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

Project Sources

Research project on Github `https://github.com/JulianSe26/aml-project`.

1.1 Kaeggle Challenge

Written by Torben Krieger

1.2 Related Work

Written by Julian Seibel

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 introduced like in [15], in which 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] [8] [11] to tuberculosis and different thoracic diseases [14] and even pulmonary [33]. Many of the contributions utilize a Convolutional Neural Network (CNN) as single predictor. However there are interesting approaches using ensemble methods like in [17], where multiple predictors assigned with weights contribute to the final prediction. This stabilizes the training process and leads to a higher overall quality of the model by utilizing different advantages of different single models.

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 were 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 fighting the pandemic and would relieve medical personal in hospitals by increasing the automation of X-ray examination. Since this is of great importance for the society, several challenges were brought to life coming up with a wide variety of solutions. In the following, we will introduce some of them that we consider suitable for our approach.

The COVID-Net [35] by Wang, Lin, and Wong uses 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 [18] network by Mangal, Kalia, Rajgopal, *et al.* In their approach, they use a pre-trained CheXNet [22] 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, in the proposed method [1] by Albahli and Albattah transformation learning is applied to fine-tune and compare three state-of-the-art models (Inception ResNetV2, InceptionNetV3 and NASNetLarge) to detect COVID-19 infections, non-COVID-19 infections (like pneumonia) and no infection. Despite the high accuracy of 99% achieved by the InceptionNetV3 model, the authors reported that all models suffer from overfitting due to limited amount of available data.

Since we are participating in the previously described challenge, we have a fixed requirement in terms of predicting not only if a X-ray image is COVID-19 positive, we rather need to detect and locate suspicious areas in such images. There exist a handful of related works we selected that try to achieve a similar goal, namely [5], [7] and [2].

Regarding the first, Brunese, Mercaldo, Reginelli, *et al.* [5] introduce a composed approach consisting of three different phases. In the first phase a image is classified as "potentially positive" by predicting if there is pneumonia patterns present in the image. The second phase then tries to differ the finding into COVID-19 caused or just pneumonia. In their last step, the authors designed a process using Gradient-weighted Class Activation Mapping to localize the affected areas that are symptomatic of a COVID-19 presence. With this approach the authors are able to add some explainability to their model by providing visually which areas are important for the final decision of the network.
The second approach by Fan, Zhou, Ji, *et al.* [7] proposed a model called Lung Infection Segmentation Deep Network (Inf-Net) that is capable of identifying and segmenting suspicious regions typical for COVID-19 infections. They use a partial decoder approach to generate a global representation of segmented maps followed by a implcit reverse attention and explicit edge attention mechanism to enhance the map boundaries. Due to the limitation of available data, the authors proposed a semi-supervised framework for training the model.
The last approach by Al-antari, Hua, Bang, *et al.* [2] a You only look once (YOLO) object detection model [23] is used to detect and diagnose COVID-19, being also capable of differentiating it from eight other respiratory diseases. In contrast to the former approach, the

proposed CAD system using the YOLO model classifies and predicts bounding boxes for regions of interest. In their paper, the authors reported a diagnostic accuracy of 97.40%.

1.3 Proposed Solution

Written by Tobias Richstein

Hier vlt. bisschen auf relevante parts aus related work eingehen + erklären dass wir pre-training machen mit den datensets die dann danach folgen

für image level: ensemble aus yolo + faster rcnn und study level dann ein model just a short introduction to our solution, models will be covered in 3 on page 5

2 Data

2.1 NIH Data

Written by Tobias Richstein

2.2 RSNA Data

Written by

Similar to the pre-training of the Faster Region Based CNN (R-CNN) backbone, we decided to utilize an additional dataset to also pre-train the YOLO model

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 [26] by Ren, He, Girshick, *et al.* is an evolutionary step in a line of 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 [10]. 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* [31] 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 [16], 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 [10] 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 [9]. 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 [28]. 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 [26]. 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*[12] 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 [12] 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 [36]. 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 [21], 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 4 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 [13] and modified slightly to provide a learning rate floor of $0.05 * \mathrm{lr}_{\mathrm{initial}}$:

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

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 $\mathrm{lr}_{\mathrm{initial}} = 0.001$ (see also figure 1).

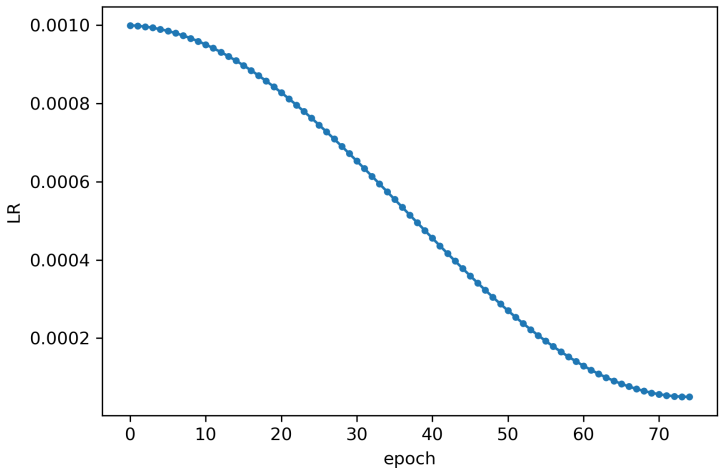

Figure 1: Learning rate schedule applied for training the detection models.

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 [19]. 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.

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 [20]. The loss numbers for the training and validation loss can be seen in 2 on the following page. 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 14.

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 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

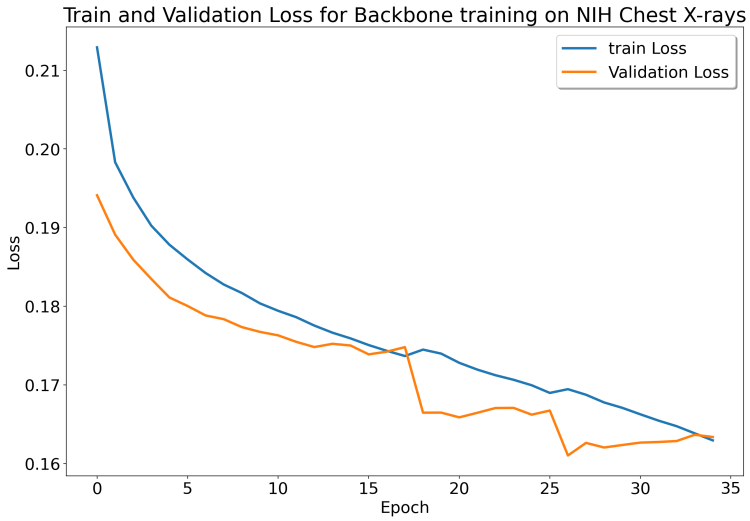

Figure 2: Loss figures of the ResNeXt training

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 3 on the following page. As will be evidenced later in chapter 4.2 on page 14 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 [9] 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 [37] and described in more detail in section 3.2 on the following page, 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 page 14.

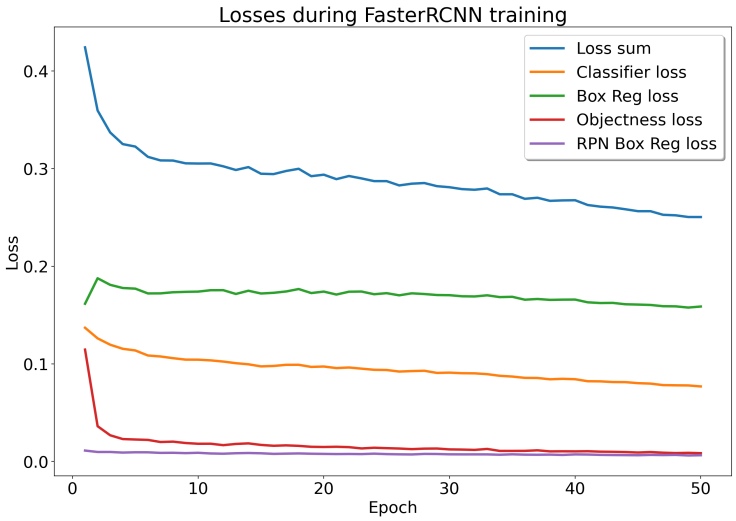

Figure 3: Loss figures of the Faster R-CNN training

3.2 YOLO

Written by Julian Seibel

The You only look once (YOLO) model originally proposed in [23] is an object detector introduced in 2015 by Redmon, Divvala, Girshick, *et al.* In contrast to the previous presented Faster R-CNN, this model makes its predictions with just a single network evaluation and is therefore called a single-shot detector (hence the name YOLO). Unlike in region proposal or sliding window based network architectures, YOLO considers the entire image for predicting which enables the model to implicitly encode context information about the objects. With this, the model is capable of learning the typical shape and size of objects, which objects are likely to occur together and what typical positions objects have in relation to other objects. The initial idea was to provide a object detection network that achieves both, high quality and high inference speed. The authors claim that their YOLO model can be up to 1000 times faster than R-CNN and up to 100 times faster than Faster R-CNN. Since its initial introduction, the YOLO model was adapted in many research problems and has been improved in several follow-up works [24] [25] to finally come up with the newest version *V4* [4]. In general, a YOLO model consist of three main pieces:

1. The backbone, similar to the Faster R-CNN, this is a deep CNN that learns image features at different angularities. In their original paper, the authors used a backbone named *Darknet* that is a neural network consisting of 53 convolution layers [25]. However this was substituted with a CSPNet [34] since *V4* [4].

2. The neck, that is a series of layers which combine features of different convolution layers. Since *V4* the PANet [30] neck is used for this part of the model.

3. The head, that part consumes the features from the neck and processes them further for the final box, confidence and class prediction.

The YOLO model divides each input image of size 512×512 into a $G \times G$ grid, where a grid cell is "responsible" for an object if it contains the objects' center point. Each grid cell predicts bounding boxes using predefined anchors and corresponding confidence scores that indicate how likely it is that the box contains an object and how well the box fits to the object. For each bounding box a confidence score is predicted using logistic regression. Unlike in R-CNN, the YOLO model does not predict an offset coordinates for predefined anchors, it rather predicts the location coordinates relative to the center of a grid cell which constraints the coordinates to fall between 0 and 1. This helps the network to learn the parameters more easy. Therefore, the model prediction for a bounding box is a quintuple (t_x, t_y, t_w, t_h, c) consisting of four coordinates and one for the object confidence (also called "objectness"), where t_x and t_y are the normalized center coordinates of the bounding box. As illustrated in figure 4, the model predicts the width and height of the bounding box as offsets from center coordinates of the box and anchors given the offset from the grid cell to the top left image corner (c_x,c_y) and the anchor width and height (p_w,p_h) [24] [25]. For

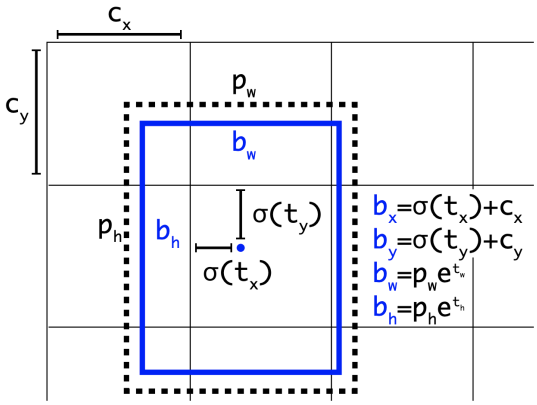

Figure 4: YOLO box prediction extracted from [25].

each predicted box a class label is assigned using a sigmoid based multi-label classification. The confidence score for a object is then the product of "objectness" and class confidence to express the probability of a certain class instance appearing in the predicted bounding box and the quality of how well the box fits the object. The model will produce outputs at different scales and depending on the number of anchors, for each grid cell multiple bounding boxes will be predicted. This often leads to a huge number of predictions per image, which is why non maximum suppression is performed to filter out boxes that do not meet a certain confidence threshold and that overlap to much in terms of IoU (used thrshold of 0.5 in our experiments).

Training of the YOLO model

For our implementation, we decided to use a model provided by [32] which is called unofficially YOLO *V5* which is based on the YOLO *V4* with some improvements in speed and quality.

We found that if we use pretrained weights on the COCO dataset for the model, we get a performance boost in terms of our selected metrics (see 4.2 on page 14 for further details). In addition, we did a pretraining for the model before we trained it to the SIIM COVID-19 dataset. For this we used the RSNA pneumonia detection dataset described in 2.2 on page 4.

We applied the same learning rate scheduling for all trainings of the YOLO model **dann training beschreiben + alle tricks - warmup, multilabel, GIOU [27]**

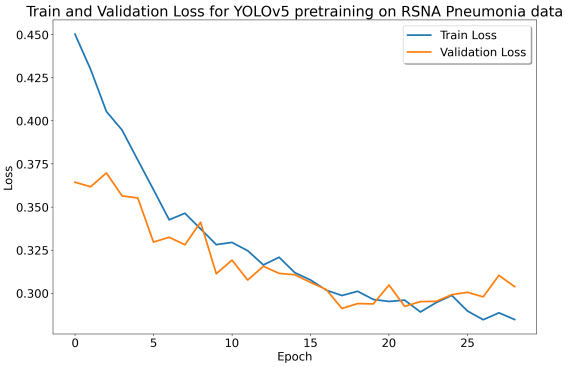

Figure 5: $dt = 0.1$

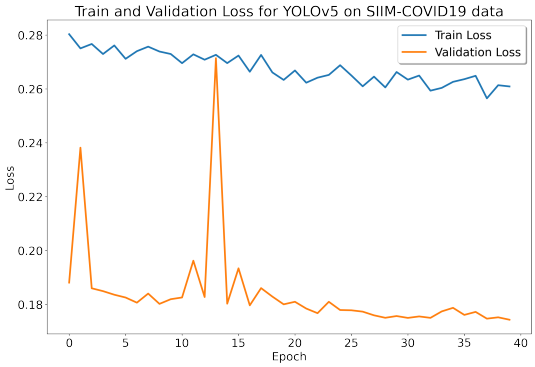

Figure 6: $dt =$

3.3 Combining detections

Julian Seibel

For our final COVID-19 detection, we decided to combine the two described models in an ensemble predictor to combine both advantages of the models and also to create a more stable bounding-box prediction considering both the outputs. For this we created a weighted box fusion following the approach of Solovyev and Wang [29], where we get a final bounding box prediction given the predictions of each model.

Each predicted box is added to a list B, that is sorted w.r.t. the corresponding confidence scores. Then two new empty lists are created for boxes clusters L and fused boxes F. Each position in F stores the fused box for the corresponding entry *pos* in L. The algorithm then iterates over the predicted boxes in B trying to find a matching box in F based on a IoU criteria (e.g. IoU $>$ Threshold, where in our case the threshold was set to 0.55). If no

match is found, the box from list B is added L and F. In contrast, if a match is found, the box is added to the cluster list L at the corresponding position to the box in F. The box coordinates (x, y) and confidence scores c in F will then be recalculated using all boxes accumulated in $L[pos]$ with the fusion equation:

$$c = \frac{\sum_{i=1}^{T} c_i}{T} \tag{2}$$

$$x_{1,2} = \frac{\sum_{i=1}^{T} c_i * x_{1,2}^i}{\sum_{i=1}^{T} c_i} \tag{3}$$

$$y_{1,2} = \frac{\sum_{i=1}^{T} c_i * y_{1,2}^i}{\sum_{i=1}^{T} c_i} \tag{4}$$

Using the confidence scores as weights, predicted boxes with higher confidence naturally contribute more to the fused box. If all predicted boxes in B are processed, confidence scores in F will be rescaled using the number of predicted bounding boxes T and and the number of participating models M:

$$c = c * \frac{min(T, M)}{M} \tag{5}$$

In our final version, we set the weights for the box fusion $w_{fusion} = (1, 1)$ meaning that each model contributes the same to the final prediction. We did not apply any training process to the ensemble model, but it would be interesting to train the ensemble approach in an end to end fashion. However, we do not have the computational resources available to jointly train both detection models.

3.4 Study-Level model

Written by Torben Krieger

4 Evaluation

4.1 Evaluation of ResNeXt

Written by Tobias Richstein

accuracy, f1, usw.

4.2 Evaluation of Faster R-CNN and YOLO

Written by Julian Seibel

4.3 Evaluation of Study-Level Model

Written by Torben Krieger

5 Proposed web application

Written by Tobias Richstein

6 Conclusion

Written by All

Future outlook and research topics /improvements:

more data (as always), E2E training for detection ensemble, explore more models (Mask R-CNN...), providing ablation studies of how pretrained weights contribute to the overall scores.

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] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. arXiv: 2004.10934. [Online]. Available: https://arxiv.org/abs/2004.10934.

[5] 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.

[6] 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.

[7] D.-P. Fan, T. Zhou, G.-P. Ji, Y. Zhou, G. Chen, H. Fu, J. Shen, and L. Shao, "Inf-net: Automatic covid-19 lung infection segmentation from ct images," *IEEE Transactions on Medical Imaging*, vol. 39, no. 8, pp. 2626–2637, 2020.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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).

[17] 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.

[18] 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].

[19] 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.

[20] 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.

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

[22] 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].

[23] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[24] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. arXiv: 1612.08242. [Online]. Available: http://arxiv.org/abs/1612.08242.

[25] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. arXiv: 1804.02767. [Online]. Available: http://arxiv.org/abs/1804.02767.

[26] 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.

[27] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union," Jun. 2019.

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

[29] 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.

[30] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[31] 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.

[32] "Ultralytics github yolo v5," 2021. [Online]. Available: https://github.com/ultralytics/yolov5.

[33] 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.

[34] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[35] 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.

[36] 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.

[37] 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.