UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Ruprecht-Karls-Universität
Heidelberg

Final project report

Advanced Machine Learning

Prof. Dr. Köthe

COVID-19 Detection on Chest X-ray images

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

With the emergence of the COVID-19 disease caused by the SARS-CoV-2 virus late 2019 [20] and the evolution to a global pandemic within 2020 the demand of proper testing methods evolved. To stop the spread of the virus and keep the occupation of isolation beds within hospitals low it is critical that the used tests are fast and reliable. Reliability relates especially to a high sensitivity as false-negative tests could lead to unexpected infections due to personal behaviour or the treatment within hospitals [53]. Nowadays the usage of Real-Time reverse transcription Polymerase Chain Reaction (RT-PCR) tests is the general case and test kits as well as laboratory capacity is broadly available in most countries [47]. At the beginning of the pandemic the availability of tests kits was generally limited [44]. In addition it was observed that the test result is highly dependent on the time of testing. According to studies there is a substantial number cases where only repeated tests after a couple of days revealed a COVID-19 infection [4].

As addition to RT-PCR several studies propose the usage of Computed Tomography (CT) scans of the chest for early and fast screening for a COVID-19 disease [13], [56]. For CT scans the analysis time within laboratory could be avoided. At the same time a chest scan could provide early traces about the severity of the upcoming disease [57].

// todo from CT to CXR

Although the American College of Radiology (ACR) not recommends to use CT or chest X-rays scans as first-line test [2] the combination with an RT-PCR test is might beneficially. Especially in cases where the initial RT-PCR tests was negative, abnormalities found within a chest X-ray could endorse a second test at a later point of time. Thus the scans could help to detect false-negatives by RT-PCR tests and avoid infections by keeping the suspicion for a COVID-19 disease. Several studies show that a detection of COVID-19 was earlier or only possible by chest X-ray or CT scans [42], [52]. However due to the exposure of radiation caused by any kind of X-ray scans the application of a COVID-19 detection by these scans

is only acceptable for a concrete suspicion with symptoms. Thus the usage of X-rays for detection is prohibitive for preventive tests.

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 [23], 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 [5] [14] [17] to tuberculosis and different thoracic diseases [21] and even pulmonary [48]. Many of the contributions utilize a Convolutional Neural Network (CNN) as single predictor. However there are interesting approaches using ensemble methods like in [26], 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 [50] 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 [27] network by Mangal, Kalia, Rajgopal, *et al.* In their approach, they use a pre-trained CheXNet [33] 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 [8], [12] and [3].

Regarding the first, Brunese, Mercaldo, Reginelli, *et al.* [8] 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.* [12] 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 implicit 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.* [3] a You only look once (YOLO) object detection model [34] 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 pretraining 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 7

2 Data

2.1 NIH Data

Written by Tobias Richstein

The first of the datasets that we used to train our models comes from the National Institutes of Health (NIH), a division of the U.S. Department of Health, and is a collection of over 100 thousand chest X-ray images by over 30 thousand patients presented by Wang, Peng, Lu, *et al.* in [51]. This dataset was published in 2017 and is therefore not concerned with COVID-19 patients at all. Rather it consists of images depicting 14 different illnesses and images of healthy patients. The authors claim that there is a large amount of data in the form of patient X-rays and corresponding findings available in hospital's archiving systems but that these have not been properly consolidated and catalogued across multiple hospitals and states before. The authors also claim that the findings are mostly embedded in sentences making them not easily machine-readable.

To overcome these issues, the authors collected the images from different hospitals and used natural language processing to extract the medical findings from the written reports associated with the images. In the initial dataset this included eight different illnesses: Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia and Pneumothorax as well as images with no finding at all. Later, the dataset was updated to the 15 class form (14 illnesses and no findings) that we use for our project to also include Consolidation, Edema, Emphysema, Fibrosis, Pleural Thickening and Hernia as findings. The class distribution can be seen in figure 1 on the next page where it is clear that the classes are very disproportionally represented. There is a total of $141,537$ diagnoses, meaning each image has an average of roughly 1.26 labels associated. This number is skewed however since if anything is found then the average goes up to 1.56 since no finding always means that only this one label is attached to a record.

The images in the dataset are given in the PNG format and are all sized 1024×1024 and with RGB color channels. The dataset does have roughly six thousand entries for bounding boxes for some of the images but those are not really of use to us since there are so few and also because the value of this dataset lies somewhere else for us: The training of our backbone network for one of the object detection networks (more on that in section 3.1 on page 7). We can use this dataset to train the backbone on images that are roughly within the problem domain that we actually want to tackle so that the feature vectors that it produces are valuable to the actual object detection network.

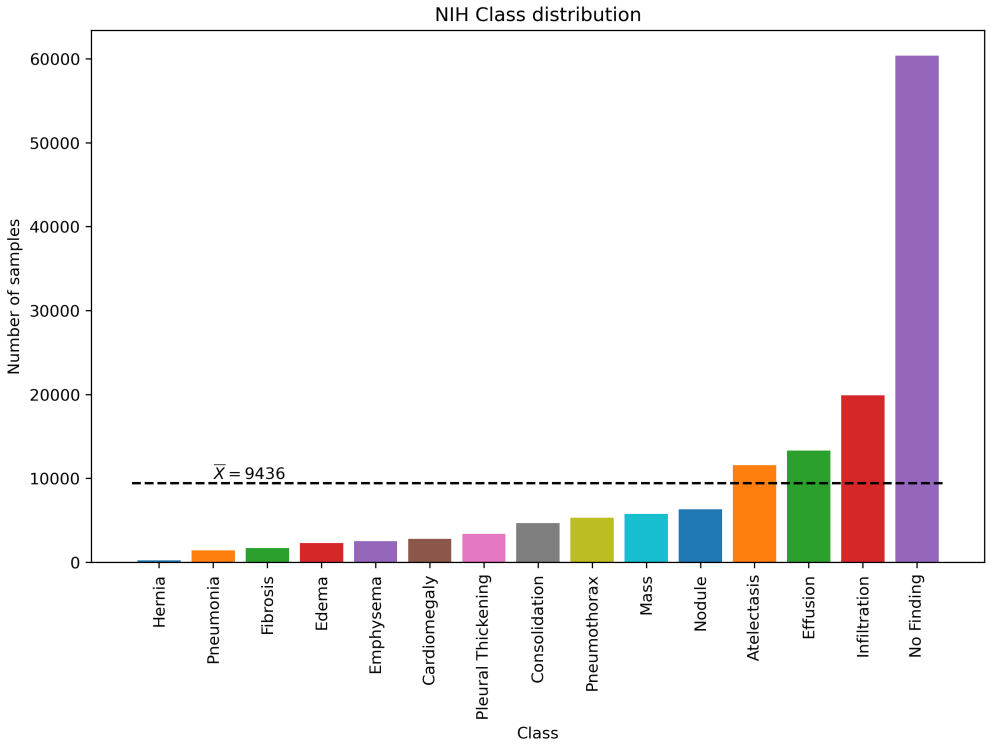

Figure 1: Class distribution in the NIH dataset

2.2 RSNA Data

Written by Julian Seibel

Similar to the pretraining of the Faster Region Based CNN (R-CNN) backbone on the above described NIH dataset, we use the RSNA dataset [22] originally published as part of a two-stage Kaggle challenge in 2018. The dataset contains $30,227$ samples from $26,684$ different patients in form of 1024×1024 DICOM images. Despite the data was anonymized, for every patient there are meta-information available including sex, age and radiographic view. The images were labeled by radiologists from Radiological Society of North America (RSNA) and the Society of Thoracic Radiology. There are three mutual exclusive labels possible (lung opacity, normal, no lung opacity/ not normal) whereas lung opacity describes a indicates evidence for pneumonia including also bounding box information for regions of interest in the format $(x_{min}, y_{min}, width, height)$. A positive image can contain multiple bounding boxes. Figure 2 shows the class distribution for the RSNA datasets. In total there are $9,555$ sample images that are labeled as positive and therefore come along with labeled regions in form of bounding boxes.

The objective of this challenge was to train a model that is capable of correct classification including also bounding box predictions for each positive case and a confidence score, indicating the model confidence for a single bounding box prediction. The final submissions

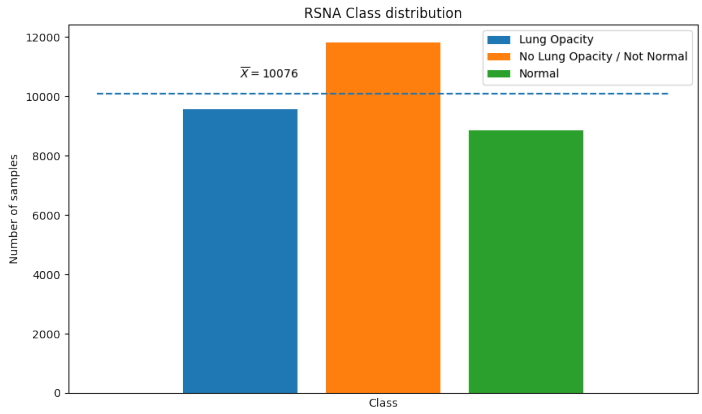

Figure 2: Class distribution in the RSNA dataset

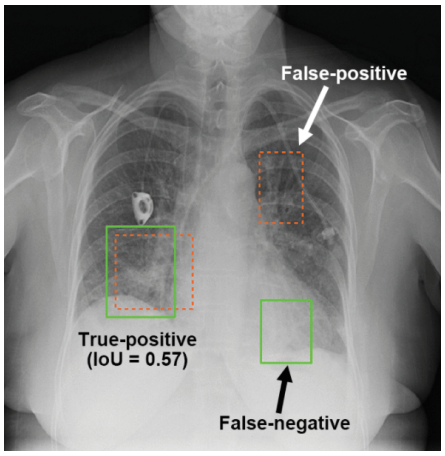

Figure 3: Extracted from [30]

were evaluated using Intersection over Union (IoU) based metrics to measure the overlap of predicted and ground truth boxes. Figure 3 shows an image from the RSNA dataset including ground truth boxes (green), possible predictions (reds-dashed) and corresponding IoU values.

Since this challenge has a lot in common with the challenge we want to contribute to in this work and the image appearance is the exact same, namely DICOM images of chest X-rays, we decided to include this dataset as additional source for pretraining our models. In contrast to the NIH dataset, the RSNA data is rather small, but brings one major advantage with labels including ground truth bounding boxes for positive pneumonia cases which makes the data suitable for pretraining object detection models. With this, we also think there are semantic similarities in predicting pneumonia and COVID-19 regions, that we hope will positively affect our final model performance for ultimately detecting COVID-19 evidence. In detail, we used the positive cases for training our detection models to "introduce" the weights into the domain of medical X-ray chest images. For the meta-data of the patients however, we did not find any suitable purpose which is why we ignored this information in the further course of this work.

2.3 SIIM COVID-19 Data

Written by Torben Krieger

3 COVID-19 Detection

3.1 Faster R-CNN

Written by Tobias Richstein

The Faster R-CNN network architecture proposed in [37] 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 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 [16]. 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* [45] 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 [24], 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 [16] 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 [15]. 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 [40]. 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 [37]. 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*[18] 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 [18] 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 [55]. 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 [32], which we did. This reference implementation has also been pre-trained on the ImageNet dataset [10], 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 (plus one for *No Finding*), 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 [19] 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 4).

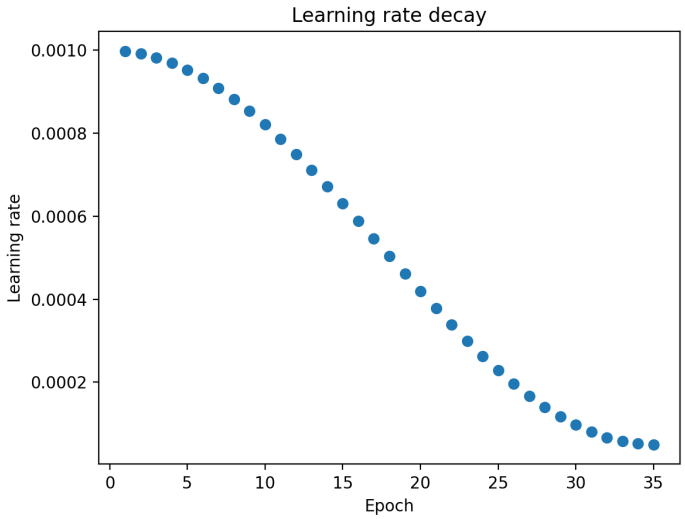

Figure 4: Exemplary learning rate schedule for 35 epochs applied during training of the detection models.

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 and essentially enlargen our dataset 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 erasing 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 [28]. 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 [31]. The loss numbers for the training and validation loss can be seen in 5 on the following page. It can be seen that in the end some overfit occurs 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 18.

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

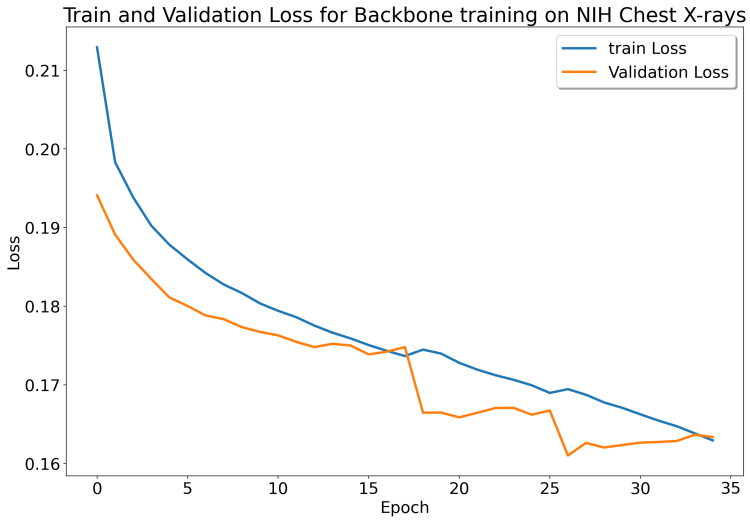

Figure 5: Loss figures of the ResNeXt training

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 6 on the following page. As will be evidenced later in chapter 4.2 on page 19 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 [15] 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 [60] 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 19.

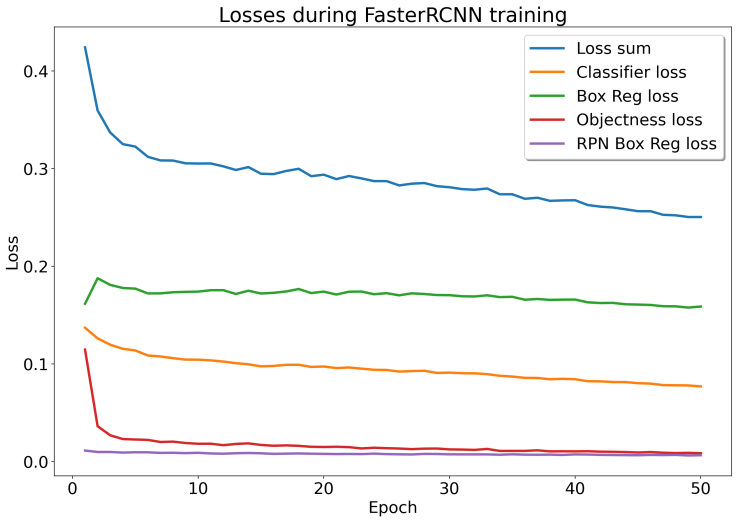

Figure 6: Loss figures of the Faster R-CNN training

3.2 YOLO

Written by Julian Seibel

The You only look once (YOLO) model originally proposed in [34] 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 [35] [36] to finally come up with the newest version *V4* [6]. In general, a YOLO model consist of three main pieces:

- 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 [36]. However this was substituted with a CSPNet [49] since *V4* [6].
- The neck, that is a series of layers which combine features of different convolution layers. Since *V4* the PANet [43] neck is used for this part of the model.

- 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 7, 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) [35] [36]. For

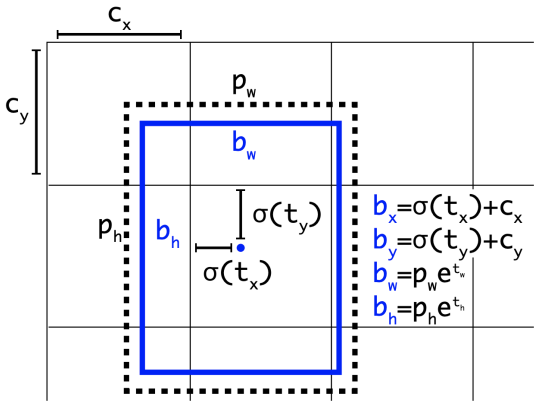

Figure 7: YOLO box prediction extracted from [36].

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 [46] which is called unofficially YOLO *V5* which is based on the YOLO *V4* with some improvements in speed and quality. For our model configuration we used the *yolov5l* model from [46].

Similar to [1] and [27], we found that there are not enough data available to achieve state-of-the-art results which is why we use a transfer-learning approach with pretrained weights on the COCO dataset, leading to a performance boost in terms of our selected metrics (see 4.2 on page 19 for further details). Furthermore, we did a second pretraining step, where we used the RSNA pneumonia detection dataset described in 2.2 on page 5, to adjust the initial COCO-weights in the direction of medical lung images by training the model for 40 epochs. Despite that the objective is different (detecting pneumonia instead of COVID-19), both datasets share many things like the prediction of just one class called *opacity*. We could see an increased performance in the evaluation if we do such a pretraining. The final model is then ultimately trained for 30 epochs on the SIIM COVID-19 data.

As already mentioned in the Faster R-CNN training, we did several experiments implementing alternative regression losses for the bounding box predictions. In the YOLO paper the authors use a mean squared error (MSE) regression loss to train the bounding box output. However, as Rezatofighi, Tsoi, Gwak, *et al.* [38] report in their work, there is not a strong correlation between minimizing the MSE and improving the IoU value. Since a plain IoU-based loss would not consider the actual distance of a prediction to a ground truth box, meaning that if there is no intersection, the IoU value would be zero and not differentiate between predictions that are close to the ground truth and predictions that are far away from the ground truth. In addition the authors concern that the evaluation of the model is mostly done using IoU-based metrics whereas the training is performed by minimizing a MSE loss.

Therefore the authors proposed to use a more appropriate loss they call Generalized Intersection over Union (GIoU) loss that uses the smallest convex hull of both bounding boxes to encode their relationship also in terms of distance. There exist several extensions like the previously described CIoU loss [60] or Distance Intersection over Union (DIoU) loss [59]. For our final model, we experienced best results using the GIoU loss proposed by Rezatofighi, Tsoi, Gwak, *et al.* The objectness and class confidence predictions were trained using a Binary Cross Entropy with Logits loss. The final loss is then a weighted sum of all the three single losses:

$$\mathcal{L}_{total} = 0.075 * \mathcal{L}_{box} + 0.05 * \mathcal{L}_{class} + 0.75 * \mathcal{L}_{objectness} \tag{2}$$

For the training process, we used a similar setup like in the Faster R-CNN training including a SGD-based optimizer with a start learning rate of $lr_{initial} = 0.01$ and momentum of 0.937, a learning rate scheduling as described in 1 and an input image size of 512×512. Because

of VRAM constrains in our hardware, we could only use a batch size of 3 (excluding augmentations) but we used a gradient accumulation based on a nominal batch size of 64, this has the effect that the optimizer and scaler only updates if the nominal batch is over rather than the limited physical batch. Furthermore we applied 1000 warm-up iterations, weight decay of 0.0005 following the approach of [46] and similar to the Faster R-CNN approach implemented the training process using *Autocasting* for speed-up.

Similar to the Faster R-CNN we applied several image transformations as augmentation technique, which include:

- Image transformations like flipping up-down (probability of 0.1) and left-right (probability of 0.5)
- Augmentation of the image and color space using different values for hue, saturation and values in the HSV space.
- Mixup [58] with a probability of 0.5

The losses for the pretraining and actual COVID-19 training are illustrated in figure 8. For the pretraining, we can see a slight increase of the validation loss after about 25 epochs. This may be caused by overfitting. On the other hand the loss for the actual training has some strange pattern: The validation loss is always smaller than the train loss, which may be caused by one of the following facts:

1. validation set to small
2. regularization/dropout which is not active during validation
3. validation loss is measured after each epoch, whereas train loss is measured during epochs, which means that there is a shift in model performance.
4. Data augmentation during training

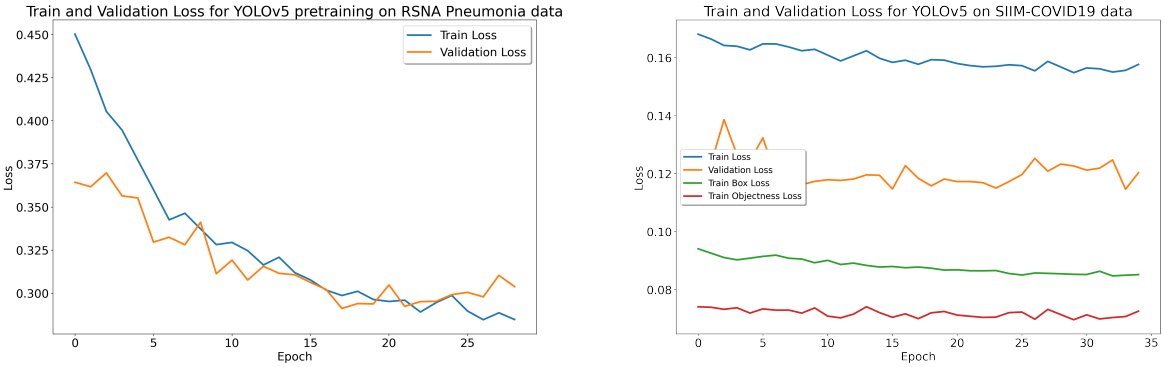

Figure 8: Train and validation loss for the pretraining on the RSNA datset (left) and the SIIM COVID-19 datset (right).

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 [41], 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:

$$c = \frac{\sum_{i=1}^{T} c_i}{T} \tag{3}$$

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

$$y_{1,2} = \frac{\sum_{i=1}^{T} c_i * y_{1,2}^i}{\sum_{i=1}^{T} c_i} \tag{5}$$

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:

$$c = c * \frac{min(T, M)}{M} \tag{6}$$

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. As fusion-criteria we set an IoU-threshold for both predictions to $IoU_{fusion} = 0.55$ which was also the best experienced parameter value described in the original paper [41]. In a last step, we do again non maximum suppression of the fusion boxes to obtain our final ensemble result. 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

The training of the ResNeXt went very well. As can be seen in figure 9 the relevant metrics of Accuracy, F1 score, Recall and precision keep going up during training. Now it is important to stress two things: First, we are not actually too interested in getting perfect results from this model in terms of classifying the 14 illnesses from the NIH dataset (from section 2.1 on page 4) and second, the number of samples is heavily unbalanced and most images have no findings at all. Especially the second fact explains the rather low recall score (ratio of true positive predictions in a class to all observations of that class) and the resulting low F1 score.

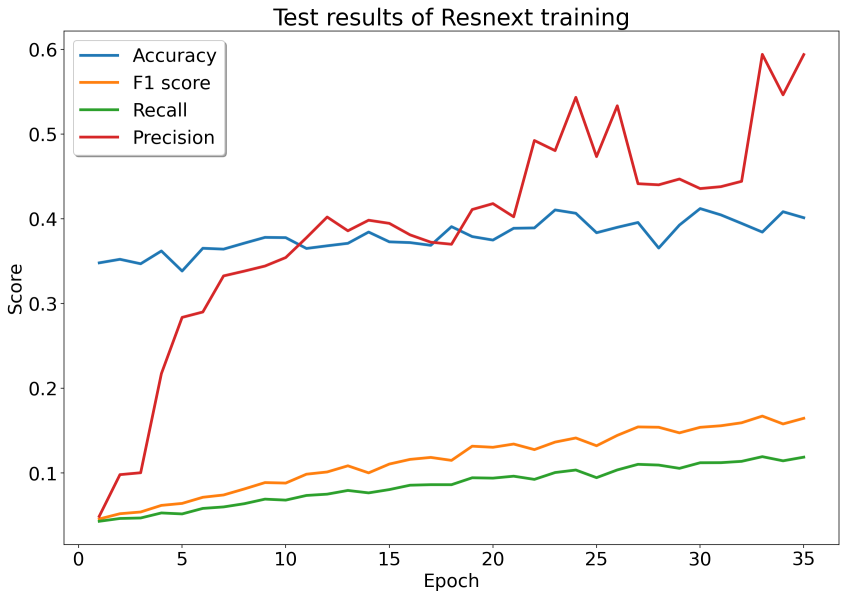

Figure 9: ResNeXt model metrics during training

In a more refined setup we could have modified the dataloader to for example over-sample all underrepresented or on the other hand under-sample some of the dominant illness labels found in the dataset to balance the training a little more. Since again we are not really interested in the actual classification but more in the feature vectors that we can use as the Faster R-CNN backbone, we are satisfied with a final Precision of 0.59, Recall of 0.12, Accuracy of 0.40 and F1 score of 0.16. We could also certainly have trained the model further

as no dramatic overfit seems to have occurred yet but as will be shown in the next section, the overall results for which we need the backbone are quite satisfactory.

4.2 Evaluation of COVID-19 detection

Written by Julian Seibel

For evaluating our COVID-19 detection capabilities, we identified two major steps that include the evaluation of the prediction quality for both the detection models, namely Faster R-CNN and YOLO and the evaluation of our ensemble approach, where we applied the method described previously in 3.3 to come up with a final prediction considering both model outputs.

There are multiple metrics possible for measuring the quality of a object detection model. However, we decided to use the widely used mean Average Precision as our crucial performance evaluation metric since many of the contributions in the field of object detection adapted the metric becoming the de-facto standard in many open opject detection challenges like Pascal VOC [11] or COCO [25]. The mAP considers both, the quality of the class prediction as well as the quality of the bounding box prediction. Regarding the latter, the IoU between ground truth and prediction is used to measure the placement quality by defining a threshold t which determines if a predicted box will be handled as true positive (IoU $> t$) or as false positive (IoU $< t$)[1]. There are various values for t conceivable depending also on the use case the model needs to fulfill. In general the threshold used is denoted as mAP@t. In our example we mainly focused on the mAP@.50 score. Hence, a predicted bounding box will be count as true positive if the IoU value indicates a 50% coverage of the ground truth. There are different ways to calculate the final mAP score, for example in [29] the authors proposed to use a all-point interpolation method to smooth the precision-recall curve to calculate the final average precision. Since our work only considers one class bounding box prediction ("opacity"), the mAP is just the average precision (AP).

First, we consider both the models and compare them in terms of our selected metrics in our validation phase that is performed after each training epoch. For the YOLO model, figure 10 shows the validation metrics during pretraining on the RSNA data. The graph shows that during the first epochs of the pretraining the metric values increase but then kind of flatten out achieving almost $\approx 60\%$ in mAP@.50. Similar to the ResNeXt backbone model, we are not particularly interested in achieving state-of-the-art scores during pretraining, rather that we experience a continuous learning for "introducing" the model and its weights to the problem domain.

[1] Since it is not necessary to count true negatives as there should just be no box, those values will be excluded also in the computation of recall and precision later on.

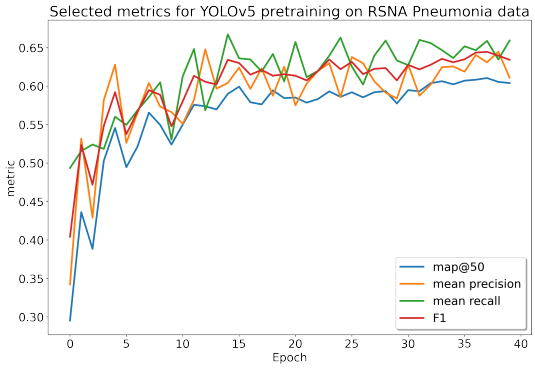

Figure 10: Validation results for the YOLO pretraining.

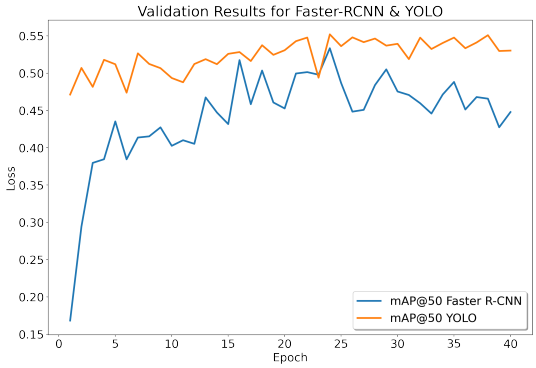

Figure 11: Validation results for YOLO and Faster R-CNN on the SIIM COVID-19 data.

The final validation results during training for both detection models on the COVID-19 dataset can be found in 11. The figure shows that we see only a slow increase in mAP@.50 for the YOLO model which is mainly caused by the pretraining. In contrast, the Faster R-CNN model shows a similar pattern like the YOLO in pretraining where with increasing number of epochs the metric values flatten out after a heavy increase in the first epochs. In general, we could observe that the YOLO model shows better results with a mAP@.50 of $\approx 55\%$, whereas the Faster R-CNN reaches $\approx 53\%$ in its best epoch. As already indicated in the training sections, we could observe a slight overfit of the model leading to a decreasing score again. This phenomenon might be caused by the limited data available or a too long training. Due to limited time capacity we did not analyzed this behavior in more detail and leave this open for further research, including hyper-parameter tuning and cross-validation methods. However, we choose the best observed models for our final implementation of the COVID-19 detection. In more detail we used the YOLO model checkpoint at epoch 26 and the Faster R-CNN model checkpoint at epoch 23.

Finally, table 1 shows extended evaluation results of the models including also the ensemble performance on a held-out consisting of 859 examples. In addition to the mAP@.50 we took F1, mean Recall and mean Precision into consideration. The results show clearly, that surprisingly, the YOLO model performs worse than the Faster R-CNN, which the validation results during training would not have let expected. However, this reflects pretty much the model performances obtained in other research, where the Faster R-CNN performs in average always better than YOLO, where the advantages in YOLO lays in the inference speed through its one-shot capability. Fortunately, we could see that combining both models through our ensemble approach leads to the best performance in all selected metrics.

To show also a qualitative comparison, figure ref shows a sample from the held-out set and the corresponding prediction of the Faster R-CNN, YOLO and Ensemble model. Figure 12

	Faster R-CNN	YOLO	Ensemble
mAP@.50	0.409	0.345	**0.451**
F1-score	0.547	0.477	**0.560**
mean Recall	0.503	0.450	**0.530**
mean Precision	0.603	0.506	**0.594**

Table 1: Final evaluation results on the held-out test set.

shows the effect of .....

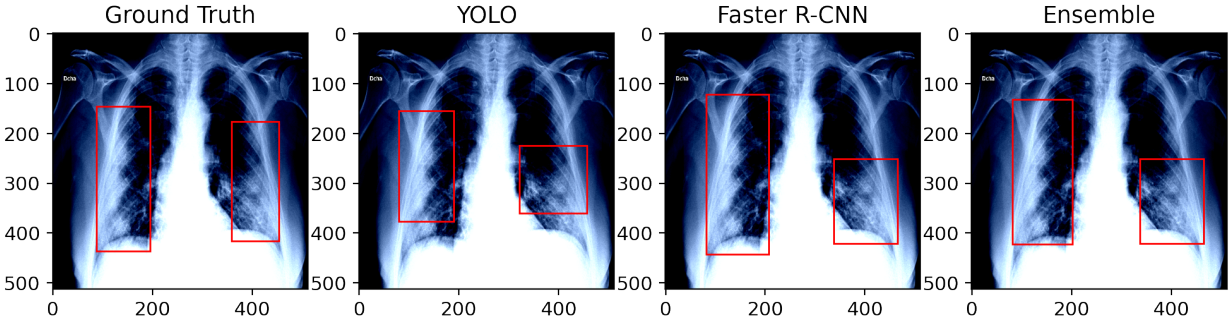

Figure 12: Prediction Results on a random example from the held-out set.

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.

vlt noch kritik an ML approaches für Corona:

- https://www.heise.de/hintergrund/Warum-KI-Werkzeuge-gegen-COVID-19-bislang-versag
  html
- https://www.turing.ac.uk/sites/default/files/2021-06/data-science-and-ai-in-the-a
  full-report_2.pdf
- https://doi.org/10.1136/bmj.m1328

In this report we have talked a lot about how different machine learning methods, and in this case computer vision, can help combat the pandemic. Honorably ML researchers and doctors all around the world scrambled to try and help at the start and during the pandemic by providing datasets, studies and models concerning all facets of what to do against the virus, for the infected or for society as a whole. Recently though, doubts have been cast on these efforts, as some studies suggest that machine learning models might not be very effective or in the worst case even dangerous. In a large scale meta study, published in the British Medical Journal, Wynants, Van Calster, Collins, *et al.* analyze 232 models proposed in 169 studies and find that not a single one is suitable for clinical use but that at least two might be promising with more work [54]. Another study by Roberts, Driggs, Thorpe, *et al.* that closely examines 62 studies regarding the topic that we were also working on of identifying COVID-19 infections in patients, comes to the same conclusion that none of them are suitable for clinical use [39]. These concerns show that while there has been great effort and a lot of good will in the research community, the processes and studies are not yet as matured as maybe ML researchers and practitioners would like. While most times no harm is done in other areas of ML usage when an error is made, the stakes in the medical field are a lot higher and therefore a lot more care has to be put into every study and ML model that is proposed.

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] American College of Radiology (ACR), *ACR Recommendations for the use of Chest Radiography and Computed Tomography (CT) for Suspected COVID-19 Infection*, Mar. 2020. [Online]. Available: https://www.acr.org/Advocacy-and-Economics/ACR-Position-Statements/Recommendations-for-Chest-Radiography-and-CT-for-Suspected-COVID19-Infection (visited on 09/09/2021).

[3] 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.

[4] I. Arevalo-Rodriguez, D. Buitrago-Garcia, D. Simancas-Racines, P. Zambrano-Achig, R. Del Campo, A. Ciapponi, O. Sued, L. Martinez-Garcia, A. W. Rutjes, N. Low, *et al.*, "False-negative results of initial rt-pcr assays for covid-19: A systematic review," *PloS one*, vol. 15, no. 12, e0242958, 2020.

[5] 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.

[6] 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.

[7] I. von Borzyskowski, Anjali Mazumder, Bilal Mateen, and Michael Wooldridge, "Data science and AI in the age of COVID-19," p. 13, Jun. 2021. [Online]. Available: https://www.turing.ac.uk/sites/default/files/2021-06/data-science-and-ai-in-the-age-of-covid_full-report_2.pdf (visited on 09/16/2021).

[8] 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.

[9] 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.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *CVPR09*, 2009.

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[12] 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.

[13] Y. Fang, H. Zhang, J. Xie, M. Lin, L. Ying, P. Pang, and W. Ji, "Sensitivity of chest ct for covid-19: Comparison to rt-pcr," *Radiology*, vol. 296, no. 2, E115–E117, 2020, PMID: 32073353. DOI: 10.1148/radiol.2020200432. eprint: https://doi.org/10.1148/radiol.2020200432. [Online]. Available: https://doi.org/10.1148/radiol.2020200432.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] C. Huang, Y. Wang, X. Li, L. Ren, J. Zhao, Y. Hu, L. Zhang, G. Fan, J. Xu, X. Gu, *et al.*, "Clinical features of patients infected with 2019 novel coronavirus in wuhan, china," *The lancet*, vol. 395, no. 10223, pp. 497–506, 2020.

[21] 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.

[22] Kaggle, "Rsna pneumonia detection challenge," 2018. [Online]. Available: `https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data`.

[23] 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.

[24] 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).

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[26] 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.

[27] 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]`.

[28] 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`.

[29] R. Padilla, S. L. Netto, and E. A. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, IEEE, 2020, pp. 237–242.

[30] I. Pan, A. Cadrin-Chênevert, and P. M. Cheng, "Tackling the radiological society of north america pneumonia detection challenge," *American Journal of Roentgenology*, vol. 213, no. 3, pp. 568–574, 2019. DOI: 10.2214/AJR.19.21512. [Online]. Available: https://doi.org/10.2214/AJR.19.21512.

[31] 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.

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

[33] 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].

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

[39] M. Roberts, D. Driggs, M. Thorpe, J. Gilbey, M. Yeung, S. Ursprung, A. I. Aviles-Rivero, C. Etmann, C. McCague, L. Beer, J. R. Weir-McCall, Z. Teng, E. Gkrania-Klotsas, J. H. F. Rudd, E. Sala, and C.-B. Schönlieb, "Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 199–217, Mar. 2021, ISSN: 2522-5839. DOI: 10.1038/s42256-021-00307-0. [Online]. Available: http://www.nature.com/articles/s42256-021-00307-0 (visited on 09/16/2021).

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

[41] 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.

[42] A. Tahamtan and A. Ardebili, "Real-time rt-pcr in covid-19 detection: Issues affecting the results," *Expert review of molecular diagnostics*, vol. 20, no. 5, pp. 453–454, 2020.

[43] 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.

[44] J. H. Tanne, E. Hayasaki, M. Zastrow, P. Pulla, P. Smith, and A. G. Rada, "Covid-19: How doctors and healthcare systems are tackling coronavirus worldwide," *BMJ*, vol. 368, 2020. DOI: 10.1136/bmj.m1090. eprint: https://www.bmj.com/content/368/bmj.m1090.full.pdf. [Online]. Available: https://www.bmj.com/content/368/bmj.m1090.

[45] 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.

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

[47] P. B. van Kasteren, B. van der Veer, S. van den Brink, L. Wijsman, J. de Jonge, A. van den Brandt, R. Molenkamp, C. B. Reusken, and A. Meijer, "Comparison of seven commercial rt-pcr diagnostic kits for covid-19," *Journal of Clinical Virology*, vol. 128, p. 104 412, 2020, ISSN: 1386-6532. DOI: https://doi.org/10.1016/j.jcv.2020.104412. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1386653220301542.

[48] 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.

[49] 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.

[50] 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.

[51] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3462–3471, Jul. 2017. DOI: 10.1109/CVPR.2017.369. arXiv: 1705.02315.

[52] Y. Wang, H. Kang, X. Liu, and Z. Tong, "Combination of rt-qpcr testing and clinical features for diagnosis of covid-19 facilitates management of sars-cov-2 outbreak," *Journal of medical virology*, 2020.

[53] C. P. West, V. M. Montori, and P. Sampathkumar, "Covid-19 testing: The threat of false-negative results," in *Mayo Clinic Proceedings*, Elsevier, vol. 95, 2020, pp. 1127–1129.

[54] L. Wynants, B. Van Calster, G. S. Collins, R. D. Riley, G. Heinze, E. Schuit, M. M. J. Bonten, D. L. Dahly, J. A. Damen, T. P. A. Debray, V. M. T. de Jong, M. De Vos, P. Dhiman, M. C. Haller, M. O. Harhay, L. Henckaerts, P. Heus, M. Kammer, N. Kreuzberger, A. Lohmann, K. Luijken, J. Ma, G. P. Martin, D. J. McLernon, C. L. Andaur Navarro, J. B. Reitsma, J. C. Sergeant, C. Shi, N. Skoetz, L. J. M. Smits, K. I. E. Snell, M. Sperrin, R. Spijker, E. W. Steyerberg, T. Takada, I. Tzoulaki, S. M. J. van Kuijk, B. C. T. van Bussel, I. C. C. van der Horst, F. S. van Royen, J. Y. Verbakel, C. Wallisch, J. Wilkinson, R. Wolff, L. Hooft, K. G. M. Moons, and M. van Smeden, "Prediction models for diagnosis and prognosis of covid-19: Systematic review and critical appraisal," *British Medical Journal*, p. 16, Apr. 7, 2020, ISSN: 1756-1833. DOI: 10.1136/bmj.m1328. [Online]. Available: https://www.bmj.com/lookup/doi/10.1136/bmj.m1328 (visited on 09/16/2021).

[55] 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.

[56] X. Xie, Z. Zhong, W. Zhao, C. Zheng, F. Wang, and J. Liu, "Chest ct for typical coronavirus disease 2019 (covid-19) pneumonia: Relationship to negative rt-pcr testing," *Radiology*, vol. 296, no. 2, E41–E45, 2020, PMID: 32049601. DOI: 10.1148/radiol.2020200343. eprint: https://doi.org/10.1148/radiol.2020200343. [Online]. Available: https://doi.org/10.1148/radiol.2020200343.

[57] R. Yasin and W. Gouda, "Chest x-ray findings monitoring covid-19 disease course and severity," *Egyptian Journal of Radiology and Nuclear Medicine*, vol. 51, no. 1, pp. 1–18, 2020.

[58] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[59] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," *CoRR*, vol. abs/1911.08287, 2019. arXiv: 1911.08287. [Online]. Available: http://arxiv.org/abs/1911.08287.

[60] 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.