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 [24] 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 [69]. 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 [62]. At the beginning of the pandemic the availability of tests kits was generally limited [58]. 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) or normal X-Ray scans of the chest for early and fast screening for a COVID-19 disease [14], [72]. For these 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 [73].

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 beneficial. 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 [56], [68]. 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.

Generally abnormalities indicating a potential COVID-19 disease, so called *opacities* could be found more easily using a CT scanner compared to normal X-Rays [25]. However normal X-Ray modalities are less expensive thus better available also within development countries. Additionally the screening is faster and simpler, the radiation exposure is less and the decontamination is easier.

Within this project we want to analyse the possibilities to detect a COVID-19 disease on Chest X-Ray (CXR) images automatically using machine learning methods. As reference we will use the "*SIIM-FISABIO-RSNA COVID-19 Detection*" Kaggle Challenge [28]. Within this challenge competitors are ask to perform object detection to identify COVID-19 typical opacities on chest X-Ray images and mark them using bounding boxes. Second a classification for a COVID-19 disease should be done. The challenge provides train data with labels and ground-truth boxes and test data without ground truth. Any submission is rated using the held back test ground truth. The images within the dataset are provided in the original DICOM format written by the used modalities. Each image belongs to a study, which relates to a single scrutiny done for a patient. A single study may contains multiple images. The dataset contains samples with or without a COVID-19 disease but also for pneumonia caused by other diseases. More details about used dataset could be found in section 2.3.

Precisely the Kaggle challenge defines following two tasks:

Image-level Perform *object detection* to detect opacities within the lung for each chest radiograph image provided. As result a bounding box should be predicted if an opacity was found. There is only a single object class (`opacity`) to be predicted. A single image could contain multiple opacities.

Study-level Predict a potential COVID-19 disease for a given study by classifying in four different classes (see below). As the prediction should be done on study level, might multiple images have to be analysed for a single classification. Although the initial challenge description denotes that multiple labels per image are possible other sources indicate that the label categories are mutual exclusive [31], [35]. Thus this task qualifies as *multi-class classification*.

It is not required that the study-level task uses the result of the image-level task as input or the other way around. The challenge only requires that the results are predicted for both.

Possible classes for the study level task are:

- Negative for Pneumonia,
- Typical Appearance,
- Indeterminate Appearance,
- Atypical Appearance.

Where the class *Negative for Pneumonia* means that no COVID or non-COVID pneumonia was detected. Contrary *Indeterminate Appearance* means that there are findings for a pneumonia but the findings are not characteristic for COVID-19. The label *Atypical Appearance* means that there were findings which are atypical or uncommon for COVID-19. Finally *Typical Appearence* means that found findings are typical for COVID-19, however other causes are still possible.

The Challenge is a so called *Code Competition*. That means competitors are asked to submit their solution as Python Notebook, not as single submission file. The notebook is then executed on cloud services provided by Kaggle. The single notebook has to prepare the data, execute the training of the models and finally create a submission file. Per submission there are limits for the total allowed runtime of the notebook. For the referenced challenge the CPU and GPU time is limited to 9 hours each [28]. Additionally no internet access is possible within the execution environment. However publicly available packages and pretrained models are allowed and accessible. The notebook has to create a single submission file which contains both the predicted labels for the image- and study-level task. Each record is identified using either the DICOM study or image id. The file has to be formatted as CSV. Results of the classification task are represented as bounding box with the size of a single pixel and a confidence of 1. For the official rating of the predicted bounding boxes a mean Average Precision (mAP) with an Intersection over Union (IoU) of 0.5 is used.

Multiple organisations joined up for hosting the challenge and providing the data, namely the *Foundation for the Promotion of Health and Biomedical Research of Valencia Region* (FISABIO), the *Medical Imaging Databank of the Valencia Region* (BIMCV), the *Radiological Society of North America* (RSNA) and as initial initiator of the challenge the *Society for Imaging Informatics in Medicine* (SIIM). The official competition ended on August 10, 2021, still late submissions are possible but the winners are already announced.

For this project we decided to take the Kaggle Challenge rather as reference but do not do a (late-)submission. This means we want to solve both tasks and conduct a validation of our results but do not intend to create a single notebook nor creating a complete submission file in the correct format.

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 [29], where the authors detect aluminium dust-induced lung diseases. With the rise of deep learning, medical image analysis on CXRs became an important subject of recent academic research reaching from detecting diseases like pneumonia [5], [15], [20] to tuberculosis and other thoracic or pulmonary illnesses [26], [64]. Many of

the contributions utilize a Convolutional Neural Network (CNN) as a single predictor. However there are interesting approaches that use ensemble methods like in [37], where multiple predictors assigned with different 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 several distinct models.

Even though it is not completely clear in which ways 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 the knowledge of previous works. Since the pandemic started in late 2019, 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 to fight the pandemic and would relieve medical personnel in hospitals by increasing the automation of CXR examination. Since this is of great importance for society, several works were published coming up with a wide variety of solutions. In the following, we will introduce some of the ones that we consider suitable for our approach.

The *COVID-Net* [66] by Wang, Lin, and Wong uses a machine-driven exploration process to design a deep convolution learning model capable of classifying CXR 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* [39] network by Mangal, Kalia, Rajgopal, *et al.* In their approach, they use a pre-trained *CheXNet* [46] CNN while substituting the output layer of the model to fit it to their requirement of predicting one of the four classes (normal, bacterial pneumonia, viral pneumonia, COVID-19). The authors applied transfer learning by only actively training the final layers of their proposed model, whereas the backbone weights are frozen. In their final comparison, the authors reported a significant improvement upon the results of the previously 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 a 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 CXR image is COVID-19 positive, but 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 [7], [13] and [3].

Regarding the first, Brunese, Mercaldo, Reginelli, *et al.* [7] introduce a composed approach consisting of three different phases. In the first phase an image is classified as “potentially

positive" by predicting whether there are pneumonia patterns present in the image. The second phase then tries to differentiate the finding into COVID-19 caused or 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 COVID-19. With this approach the authors are able to add some explainability to their model by providing visualizations of areas important to the final decision of the network.
The second approach by Fan, Zhou, Ji, *et al.* [13] proposes 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 an implicit reverse attention and explicit edge attention mechanism to enhance the map boundaries. Due to limited available data, the authors propose a semi-supervised framework for training the model.
In the last approach by Al-antari, Hua, Bang, *et al.* [3] a *You only look once (YOLO)* object detection model [47] is used to detect and diagnose COVID-19, with the model also being capable of differentiating it from eight other respiratory diseases. In contrast to the previous approach, the proposed CAD system uses the YOLO model to classify and predict bounding boxes for regions of interest. In their paper, the authors report a diagnostic accuracy of 97.40%. Going a step further, Podder, Bhattacharjee, Roy, *et al.* use a mask based prediction model (*Mask Region Based CNN (R-CNN)*) to train and test a classifier that distinguishes between patients infected and not infected with COVID-19. In this paper, the authors achieve an accuracy of 96.6% evidencing the huge potential of Mask R-CNN as a suitable model candidate [43].

1.3 Proposed Solution

Written by Tobias Richstein

To tackle the described Kaeggle Challenge, we propose a multi-faceted approach to the solution. Other works in this field have shown that using CNNs can be a viable strategy to detect a multitude of illnesses in X-ray images and in some cases also locate them. Another approach described above is to just classify the image but not to provide bounding boxes. In accordance with the Kaeggle challenge, we need to classify the given X-ray images and provide bounding boxes for suspicious areas, by means of object detection, that might be indicators of a COVID-19 pneumonia.

Object detection in images is largely dominated by four network architectures: RetinaNet [32], Single Shot MultiBox Detector [36], Region Based CNN [17] and YOLO [47]. As shown in the Related Work section, the latter two have been shown to be suitable for the task at hand in [43] or [3] respectively which is why we decided to also focus on these two architectures for our object detection solution at the image level. We use a Faster R-CNN with a ResNeXt backbone that we pre-train on the NIH-dataset as well as a YOLO network whose backbone is pre-trained on the RSNA dataset. Pre-training the backbone helps the

actual detector because the feature vectors that the backbone generates are situated within the actual problem domain instead of being generalized towards a dataset like ImageNet. The actual training for the object detection of opacities is done on the dataset of the Kaeggle challenge. We use an ensemble box fusion to combine the bounding boxes of both networks to generate more accurate results and achieve respectable results on a held out validation set at the image level.

At the study level which consists of one or more images of a patient we employ a classification network which reuses the same ResNeXt backbone as the Faster R-CNN to draw conclusions about whether a patient study shall be looked at as COVID-19 positive. **ELABORATE MORE**

To make the solution a bit more approachable and also allow some hands-on experience for users not necessarily familiar with how to run inference on their own, we built a prototype web-app to which images can be uploaded and the different models can be tested. It allows users to see the predictions of each model, image and study level, and also how the box fusion boosts the detection accuracy.

The datasets we use and why we use them is described in the next section 2 on the following page and our model approaches are detailed in section 3 on page 15. Later on we evaluate our approach in section 4 on page 29 and describe the web-app in section 5 on page 35 before coming to a conclusion later in section 6 on page 37.

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 [67]. 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 CXRs 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 15). 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.

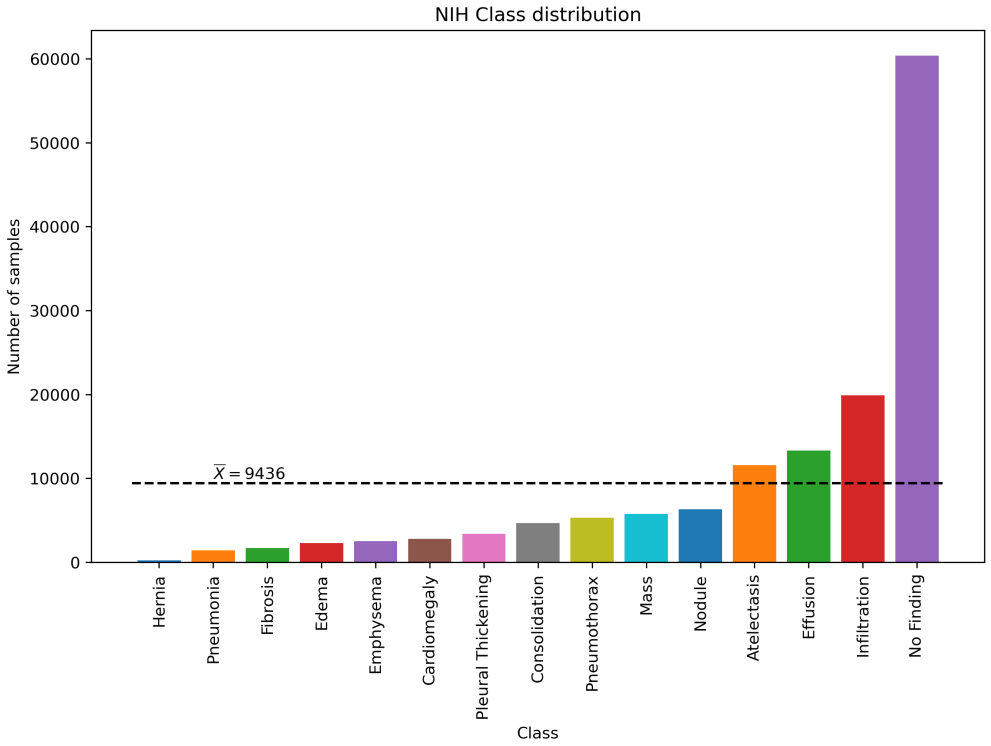

Figure 1: Class distribution in the NIH dataset

2.2 RSNA Data

Written by Julian Seibel

Similar to the pre-training of the backbone on the above described NIH dataset, we use the Radiological Society of North America (RSNA) dataset [27] originally published as part of a two-stage Kaeggle challenge in 2018. The dataset contains $30,227$ samples from $26,684$ different patients in the form of 1024×1024 DICOM images. Despite the data being anonymized, there is some metadata available for every patient including sex, age and radiographic view. The images were labeled by radiologists from the RSNA and the Society of Thoracic Radiology. There are three mutually exclusive labels possible (lung opacity, normal, no lung opacity/ not normal) whereas lung opacity indicates evidence for pneumonia including also bounding box information for regions of interest in the format $(x_{\max}, y_{\min}, \text{width}, \text{height})$. An image that is labeled positively can contain multiple bounding boxes. Figure 2 shows the class distribution for the RSNA dataset. In total there are $9,555$ sample images that are labeled as positive and therefore come along with labeled regions in the 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

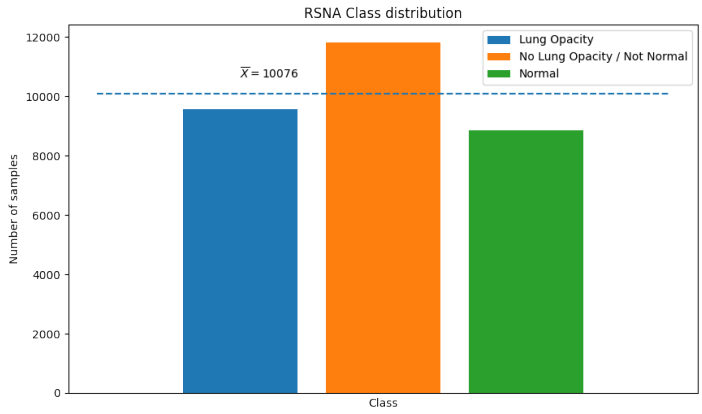

Figure 2: Class distribution in the RSNA dataset

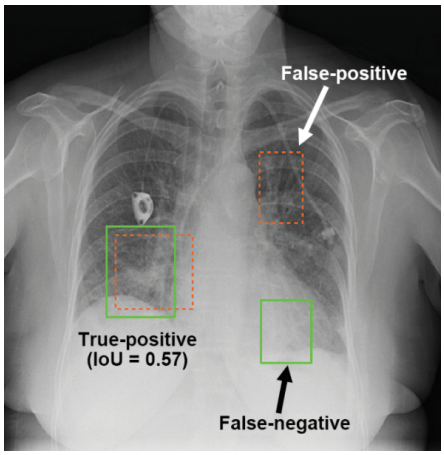

Figure 3: Extracted from [42]

were evaluated using 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 (red-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 CXR, we decided to include this dataset as an additional source for pre-training 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 pre-training 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 symptoms. In detail, we use the positive cases for training our detection models to "introduce" the weights to the domain of medical X-ray chest images and the task of finding possible regions of interest evidencing typical patterns of a certain disease. For the metadata of the patients however, we did not find any suitable use which is why we ignored this information in the further course of this work.

2.3 SIIM COVID-19 Data

Written by Torben Krieger

The last dataset we use is provided by the SIIM Kaggle Challenge itself [28]. It contains CXR images of COVID-19 positive and negative patients. The dataset itself is a reassembly of at least two previously published datasets. Namely the *BIMCV-COVID19 Data* initially published by Vaya, Saborit, Montell, *et al.* in June 2020 [63] and the *RSNA MIDRC-RICORD Data* published by Tsai, Simpson, Lungren, *et al.* in January 2021 [59]. Only the X-ray images

were taken from both datasets the annotation of the data was recreated for the assembled dataset. Both datasets were refreshed and supplemented multiple times. The description of the Kaggle Challenge not specifies which versions were used for the assembly.

The initial version of the *BIMCV-COVID19* dataset contains 1,380 CR images and 885 DX images [63]. Generally we do not distinguish between the two classical X-ray modalities CR and DX, thus the dataset contains 2265 CXR images. The images were taken from 1,311 unique COVID-19 positive patients in hospitals within the region of Valencia, Spain. All images are provided in the DICOM format including metadata, although anonymized for privacy reasons. Nevertheless patients age and sex was retained. The original dataset includes additional data, e.g. the results of one or multiple COVID-19 tests (at least one positive) and a corresponding radiological report.

The *MIDRC-RICORD* dataset contains 1000 CXR images of 361 unique COVID-19 positive patients [59]. Where positive means that at least one RT-PCR test was positive for the corresponding patient. The data was collected from four different hospitals in Istanbul, San Francisco, Toronto and São Paulo and is provided in DICOM format including metadata, again anonymized. Also here patient sex and age was retained, the authors specify that 148 out of the total amount of patients is female (41%).

As stated above the Kaggle dataset is a combination of the datasets described before. As both datasets contain CXR images for positive tested patients only, there has to be at least one additional data source. Unfortunately the description of the challenge does not provide any details about this or the exact collocation of the assembled dataset. The new dataset was annotated by professional radiologists. Per image opacities were marked with bounding boxed if found. On study level a classification into the categories, as described in section 1.1, were done by the radiologists. The classification follows the schema described by Litmanovich, Chung, Kirkbride, *et al.* where each class is mutual exclusive [35]. It is crucial to emphasize that radiologists had access to the COVID-19 status while creating the annotations [31].

The dataset consists out of a train and test set, however the test labels are not published. A test is only possible by submitting a solution as notebook. As we decided that we will not hand in our results, and use supervised methods only, we are limited to the train set for this report. The train set contains 6,334 images, all images are provided in the DICOM format. Metadata is available but anonymized. Out of the patient data only the patient sex was retained. All DICOM image files of the train set have a uncompressed size on disk of 108.2 GB after the download. The images belong to 6054 studies. The count of unique patients could not be extracted from the anonymized metadata and is not specified. The resolution of the images reaches from 846 to 4,891 pixel rows and 1,228 to 4,891 pixel columns. All images are single channel greyscale images. The DICOM standard defines different representations for image pixel data, so called *Photometric Interpretations* [10]. Within the dataset we could find 4,633 images with the `MONOCHROME2` representation, which means that the minimum pixel

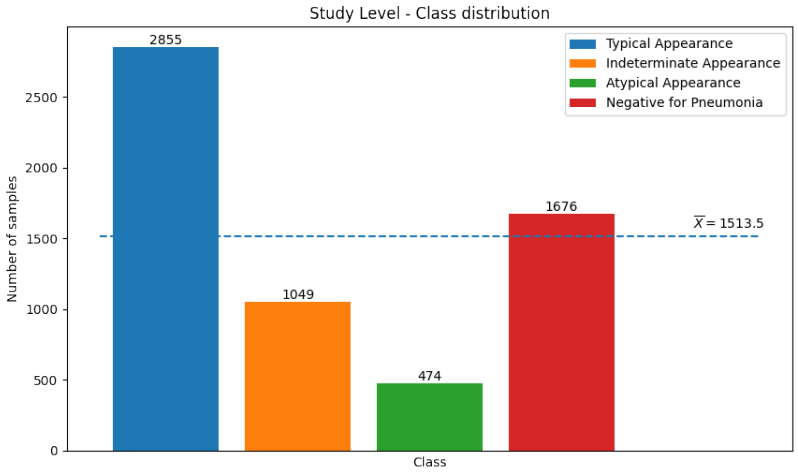

Figure 4: Distribution of classes for studies within the train dataset.

value should be interpreted as black. Contrary 1,701 images are encoded with `MONOCHROME1` representation, minimum pixel value means white color here. In DICOM modalities could store a custom amount of bits per pixel value, which relates to the bit depth of the greyscale channel in our case. Most images are captured with an bit width of 12 bits, in total 4,311 samples. All other images have a bit width between 8 (401 occurrences) and 16 bits. For training it is required to convert the pixel data of all files to a single representation and bit depth. Specifically by reducing the bit depth we lose some information which is might relevant for good predictions. Most of the included DICOM files contain uncompressed pixel data, only 440 files hold the pixel data compressed using the JPEG Lossless format. As expected the DICOM metadata field specifying the examined body part indicates "CHEST" most often, although encoded in different languages (e.g. "THORAX", "PECHO"). However some unexpected values are included, e.g. 57 files claim "SKULL" as examined body part. We checked some of these files manually, it seems that this is rather a misconfiguration of the modality. All images checked have shown the chest of the screened patients. For the complete train dataset 2,770 images were taken of female patients, 3,564 of male patients.

Most of the 6,054 studies were classified as *Typical Appearance* (2,855 occurrences). In contrast to this the class *Atypical Appearance* has only 474 occurrences, which is roughly a sixth of most frequent class. Thus the dataset is imbalanced with respect to a domination of potential COVID-19 positive samples. Figure 4 shows the complete class distribution within the train set. Per description of the challenge for the classes *Typical* and *Indeterminate* bounding boxes marking opacities should be present always [31]. Contrary to that for the negative class no bounding boxes were created. For *Atypical Appearances* sometimes a bounding box is included. In total we found 2,040 images without any bounding box / opacity. Figure 5 shows an overview on the count of boxes per image. Most were annotated

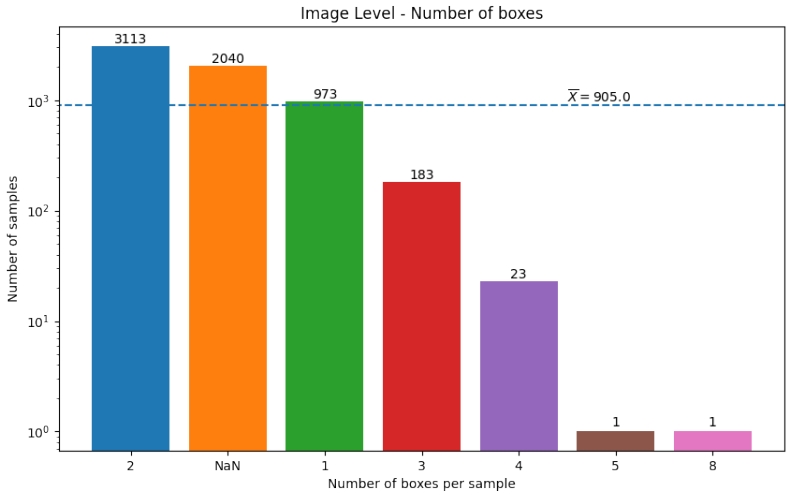

Figure 5: Count of boxes for opacities found per image. Where *NaN* means no box / opacity.

with two opacities. The figure 6 prints the number of boxes over the box size, thus giving an impression on the box sizes within the images. Related to this the description says that the radiologists opted for a single larger bounding boxes when multiple adjacent opacities were found, rather than creating multiple smaller boxes [31]. Bounding boxes for the image level and classes per study are provided in two separate CSV files.

Already by the construction of the dataset itself we could foresee potential issues with regards to the real generalization capability of trained models. Most importantly the dataset is constructed out of several sources, where specific sources provided positive or negative samples only. Thus it could be that a potential solution learns shortcuts introduced by the processes or equipment used by a specific image source. An example could be burned in image markers at a specific position of the X-ray. We were able to find some re-occurring burned-in markers at the same position for a couple of images within the dataset. The next problem are the facts we do not know about the dataset, e.g. the age distribution or the position of the patient while retrieving the X-ray scan. For example one could imagine that images retrieved in a lying position correspond to more severe and potential positive cases. All the critical points described before were mentioned within a paper by DeGrave, Janizek, and Lee [8] this paper explicitly investigated the *BIMCV-COVID19* dataset and found the aforementioned shortcomings therein. As the Kaggle dataset is a reassembly of this dataset we have to expect that these issues are also present within it. Additionally we only have very limited information about the construction process of the dataset. Summing up a comprehensive conclusion about the generalization capability of our methods will not be possible by using this dataset.

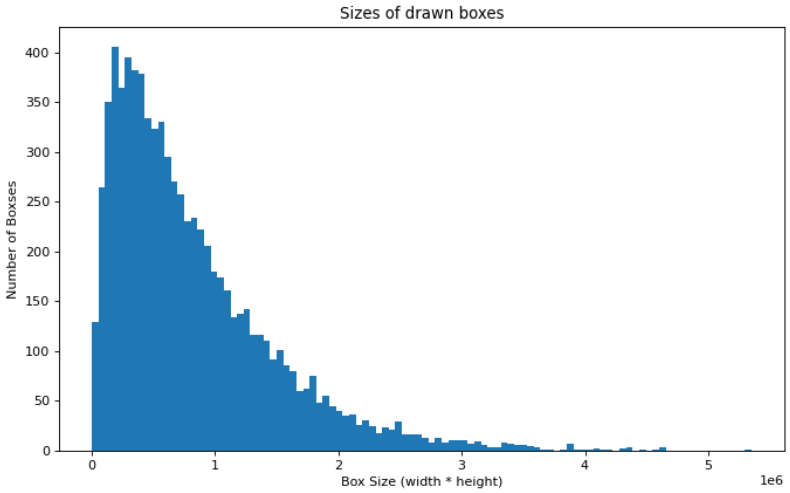

Figure 6: Count of boxes over box sizes for found opacities.

Preprocessing of the dataset & COCO

Independent of the applied model or task we applied a first preprocessing to the dataset. In general we apply the following to each DICOM image file:

- Applying VOI-LUT, if present within DICOM metadata. This is a necessary pixel value transformation for some modalities to transform pixel values retrieved by the X-ray detector to a specific representation for displaying the image [10]. Not all DICOM files come with a VOI-LUT attribute.
- Inverting pixel values for `MONOCHROME1` images. By doing so we make sure that all pixel values within the images comply to the `MONOCHROME2` representation, meaning the minimum value is interpreted as black.
- Rescale all greyscale pixel values to a value range of $[0, 255]$ to represent them with a bit depth of 8 bit.
- Store each image as lossless compressed grayscale image using the Portable Network Graphics (PNG) file format.

By applying the above mentioned preprocessing steps we reduced the overall size of the dataset to 25 GB. We also dropped all DICOM metadata as we do not require these information for the training and evaluation process. Still it is worth to mention again that the rescale of all image pixel data to a bit depth of 8 bit implies a significant loss of image information. However as we require to use pre-trained models for this project this is definitely required.

For the object detection task on image level we need a specific format of annotation to make use of several pre-trained models. This format is specific to the Microsoft Common Objects in Context (COCO) dataset initially described by Lin, Maire, Belongie, *et al.* in 2015. This dataset is one of the largest and commonly used dataset for object detection, segmentation and captioning. Due to that fact most pre-trained models for these tasks were trained using this dataset. To reuse these pre-trained models easily for other tasks the custom dataset has to comply to the label format used within the COCO dataset.

```
{
  "images": [
    { "id": int, "width": int, "height": int, "file_name": str,
       "date_captured": datetime }, (..)
  ],
  "annotations": [
    { "id": int, "image_id": int, "category_id": int, "area":
       float, "bbox": [x,y,width,height] }, (..)
  ],
  "categories": [
    { "id": int, "name": str }, (..)
  ]
}
```

Listing 1: Basic COCO annotation format in [33].

The label format is defined as JSON format, listing 1 shows the most important content nodes and attributes for our task. The node `images` contains per image one entry, the `file_name` attribute could be a relative path. The node `annotations` contains a single node per bounding box for any image. We provide functionality to convert the image level CSV file provided by the Kaggle dataset into a single JSON file which conforms to the COCO format.

3 COVID-19 Detection

3.1 Faster R-CNN

Written by Tobias Richstein

The Faster Region Based CNN (R-CNN) network architecture proposed in [50] 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 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 [17]. 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* [60] 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 [30], 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 [17] 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 [16]. 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 [54]. 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 [50]. 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*[22] 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 [22] 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 [71]. 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 [45], which we did. This reference implementation has also been pre-trained on the ImageNet dataset [9], 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 7 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 [23] 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 7).

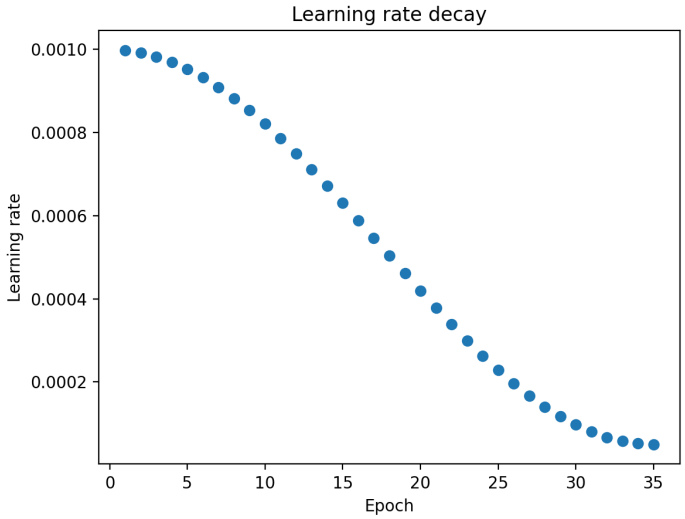

Figure 7: Exemplary learning rate schedule for 35 epochs applied during training of the detection models.

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 [40]. 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 [44]. The loss numbers for the training and validation loss can be seen in 8 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 29.

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.

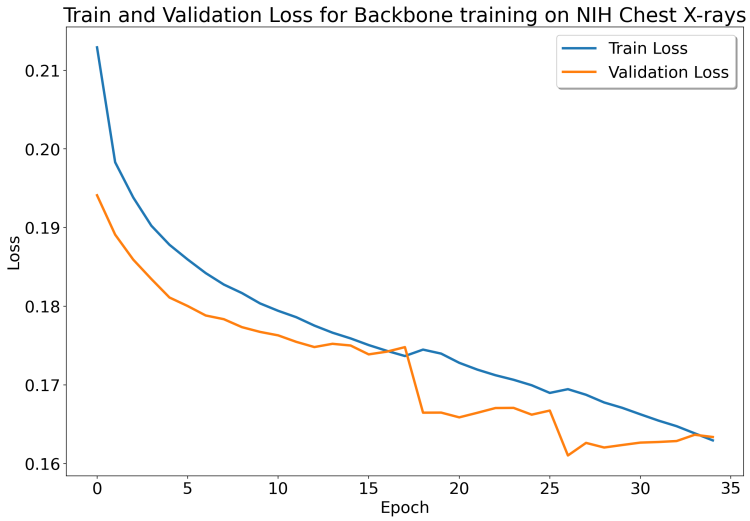

Figure 8: Loss figures of the ResNeXt training

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 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 9 on the following page. As will be evidenced later in chapter 4.2 on page 30 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 [16] 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 [76] 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

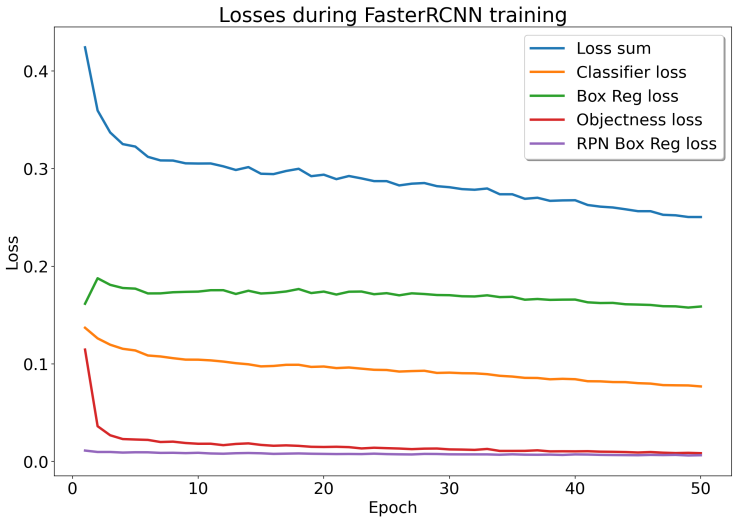

Figure 9: Loss figures of the Faster R-CNN training

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 30.

3.2 YOLO

Written by Julian Seibel

The You only look once (YOLO) model originally proposed in [47] is an object detector introduced in 2015 by Redmon, Divvala, Girshick, *et al.* In contrast to the previously 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 when 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 an 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 [48] [49] to finally come up with the newest version *V4* [6]. In general, a YOLO model consists 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 [49]. However this was substituted with a *CSPNet* [65] since *V4* [6]
- The neck, which is a series of layers that combine features of different convolution layers. Since *V4* the *PANet* [57] neck is used for this part of the model
- The head, that consumes the features from the neck and processes them further for the final box, confidence and class predictions

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 object's 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-CNNs, the YOLO model does not predict offset coordinates for predefined anchors, but rather predicts the location coordinates relative to the center of a grid cell which constrains the coordinates to fall between 0 and 1. This helps the network to learn the parameters more easily. 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 10, 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) [48] [49].

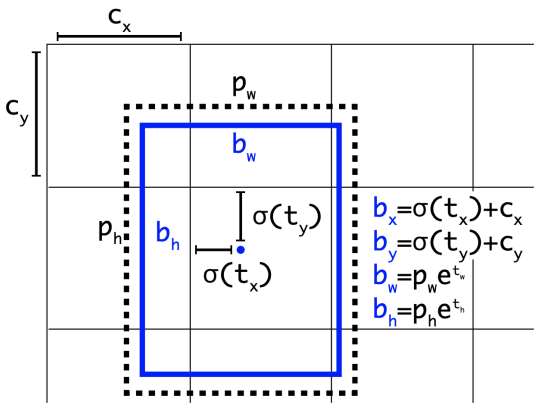

Figure 10: YOLO box prediction extracted from [49].

For each predicted box a class label is assigned using a sigmoid based multi-label classification. The confidence score for an 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 on 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.

Training of the YOLO model

For our implementation, we decided to use a model provided by [61] which is unofficially called YOLO *V5* which is based on the YOLO *V4* with some improvements of speed and quality. For our model configuration we use the *yolov5l* model according to [61].

Similar to [1] and [39], we found that there is not enough data available to achieve state-of-the-art results which is why we use a transfer-learning approach with pre-trained weights on the COCO [34] dataset, leading to a performance boost in terms of our selected metrics (see 4.2 on page 30 for further details). Furthermore, we did a second pre-training step, where we used the RSNA pneumonia detection dataset described in 2.2 on page 8, to adjust the initial COCO trained weights in the direction of medical lung images by training the model for 30 epochs. Despite the objective being 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 when we did such a pre-training. The final model is then ultimately trained for 35 epochs on the SIIM COVID-19 data.

As already mentioned in the Faster R-CNN training section, 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.* [51] 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 put forth 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 propose to use a more appropriate loss which 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 are several extensions like the previously described CIoU loss [76] or Distance Intersection over Union (DIoU) loss [75]. 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 an SGD-based optimizer with a starting learning rate of $lr_{initial} = 0.01$ and momentum of 0.937, a learning rate schedule as described in equation 1 and an input image size of 512×512. Because of VRAM constraints in our hardware, we could only use a batch size of three (excluding augmentations) but we used a gradient accumulation based on a nominal batch size of 64, which has the effect that the optimizer and scaler only update if the nominal batch is reached rather than the limited physical batch. Furthermore we applied 1000 warm-up iterations, weight decay of 0.0005 following the approach of [61] and similar to the Faster R-CNN approach implemented the training process using autocasting for speed-up. After a forward pass we reduce the amount of possible bounding boxes using non maximum suppression with a confidence threshold of 0.1 and an IoU threshold of 0.2. Similar to the Faster R-CNN we apply 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 [74] with a probability of 0.5

The losses for the pre-training and actual COVID-19 training are illustrated in figure 11. For the pre-training, we can see a slight increase of the validation loss after about 25 epochs where we assume it may be caused by overfitting (also visible in the corresponding validation metrics described in 4.2). The right side of the figure shows the loss and its components for the actual training on the COVID-19 data. It can be seen that also here the loss decreases over time, starting already at relatively low loss values. This and the fact that the losses do not decrease as steeply as in the previous training is mainly caused by the transfer learning approach using the pre-trained weights on the RSNA data. Additionally, the loss graph shows some uncommon patterns which we faced in all our experiments using the YOLO model: The validation loss is always smaller than the train loss. Since we did not see any break-in of model performance in terms of metrics measured, we did not follow-up on this, since we could exclude any wrong learning or restrictions of the model. However, such patterns may be caused by:

- The validation set being too small
- Regularization, e.g. dropout that is not active during testing or validation

- Some shifts due to the time of measurement, the train loss is obtained after every epoch whereas the validation is only measured after an epoch completes. This could cause the mean being pulled from bad performance at the very first iterations of each epoch
- Data augmentation which is only used during the training process but not included in the validation or testing process

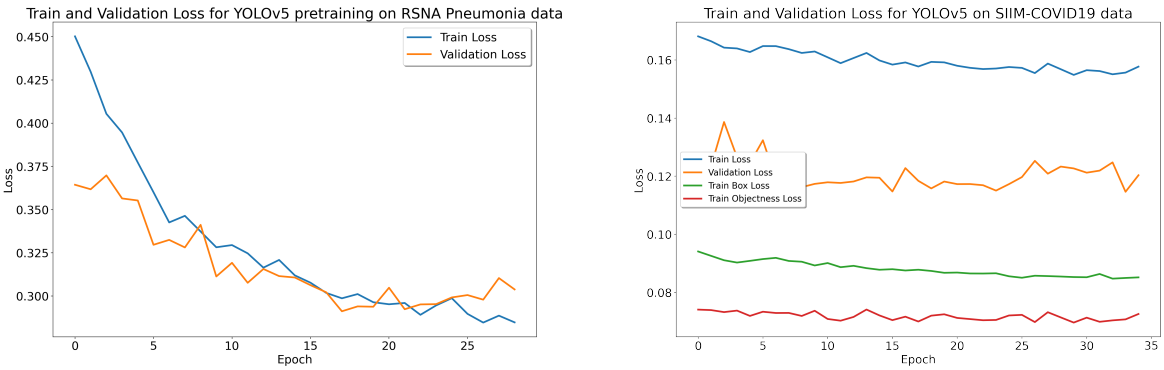

Figure 11: Train and validation loss for the pre-training on the RSNA datset (left) and the SIIM COVID-19 datset (right).

3.3 Combining detections

Julian Seibel

For our final COVID-19 detection, we decided to get the two described models in an ensemble predictor to combine both advantages of the models and also to create a more stable bounding-box predictor that considers both outputs. Following the approach of Solovyev and Wang [55], we created a weighted box fusion, where we get a final bounding box prediction given the predicted boxes and confidence scores from the YOLO and Faster R-CNN models.

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 box 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 an IoU criterion (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 to 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 re-scaled 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 [55]. 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

As described in 1.1 the study level task could be seen as multi-class classification. A study consists out of one or more CXR images, thus we could interpret the task as image classification and combining the results of each individual image afterwards. However we decided to skip this combination and perform the classification for single images only. Besides the fact that only a few number of studies consist out of multiple images, a moderator of the challenge reported within a discussion thread that most often the reason for multiple images are quality issues within the other images(s) of the study [1].

[1] https://www.kaggle.com/c/siim-covid19-detection/discussion/240250#1321539

Image Classification using ResNeXt

A CNN is the proper neural network structure to perform classification tasks on images. As the amount of training data available for the Kaggle Challenge is quite limited, we have to use a pre-trained model to utilize a recent CNN network architecture for this task. Most available pre-trained models are trained on the ImageNet dataset, which consists out of real world images [9]. Thus we consider that a re-training of the classification layer only is not sufficient. However we believe that even for a simple re-adaption of the feature extraction layers the provided dataset is too small. Consequently we think a second pre-training of the model using the NIH Dataset, as introduced in 2.1, is necessary also for this task.

Initially we thought about using the output of our object detection model as additional input or hint for the image classification model. For example as additional input channel using a mask or by a separate feature extraction model feeding into the same classification layers. However we came to the conclusion that we will not use the output. On the one hand the study level model would somehow inherit the errors of the detection model. On the other hand one could argue that the model it self should be capable to learn characteristic like opacities which indicate a potential COVID-19 disease. Or in other words, the image classification model should be able to learn the same semantics as the object detection model when using the same data.

For simplification and to avoid redoing the pre-training of the model on the NIH dataset we decided to reuse the trained backbone model of the Faster R-CNN, as described in 3.1. Therefore the weights of the trained `ResNeXt 101 32x8d` model are taken over and the classification layer is replaced. The new classification layer consists out of two fully connected layers intersected by a $ReLU$ activation function. The pooled output of the feature extraction layers have a dimensionality of 2048, which we reduced to 1024 with the first fully connected layer. The final layer consists out of 4 neurons, each representing a single class. As the *ResNeXt* was trained on ImageNet initially it requires RGB images with a size of 224×224 pixels as input [71].

Training of the ResNeXt for Image Classification

For taking over the weights of the trained backbone *ResNeXt* a new model was defined using PyTorch. This model consists out of the same pre-built and trained `ResNeXt 101 32x8d` and the above mentioned classification layers. The classification layers were initialized by *Xavier Initialization* [18]. Afterwards a checkpoint of the trained backbone model was loaded into the model with non-strict policy. This means even if parameters of the model are not contained within the checkpoint the import will not fail, which is required as the new classification layers are not part of the checkpoint.

Due to the usage of the *ResNeXt* the same data augmentation as described in 3.1 is required. Additionally we applying the same training augmentations to the SIIM dataset for the study level training. The pre-training on the NIH dataset was done as multi-label classification. As the Study level task implies a multi-class classification task other normalization and loss functions are required. While the training of the *ResNeXt* on the SIIM dataset we use *Softmax* to normalize all outputs to a sum of 1 [19]. As loss function we use *Cross Entropy*, again for numerical stability we apply the combined PyTorch loss function working on logits. Apart from this the initial setup was similar to the backbone training. This means SGD was used as optimizer and *Cosine Annealing* for learning rate decay scheduling.

After a first training run we noticed that even for an decreasing training loss the validation loss starts to increase after few epochs. We identified this generalization error as overfitting and tried to apply following regularization techniques as countermeasure (both separately or combined):

- Adding a *Dropout* layer between the last two fully connected layers and validating different probabilities for dropping output values with $0.5 \leq p \leq 0.7$
- Applying additional train data augmentations or executing them with higher probabilities.
- Using *weight decay* to applying *L2 regularization*

Weight decay was applied as defined in equation 7, where $\boldsymbol{\theta}$ represents all weights, α the learning rate and λ the rate of decay each step. Effectively weight decay means that existing weights are decreased exponential each update step. Thus the value of the weights decay to zero if not balanced by the weight update due to the gradient step. The method is comparable to adding a *L2 regularization* term to the loss function when using *SGD* [38] and penalizes large weights in practice.

$$\boldsymbol{\theta}_{t+1} = (1 - \lambda)\,\boldsymbol{\theta}_t - \alpha \nabla \mathrm{Loss}_t\,(\boldsymbol{\theta}_t) \tag{7}$$

We tried several *decay rates* in a range between $[0.00001, 0.3]$ but were unable to regularize the overfitting successfully thereby. Figure **??** shows the training loss for some of the validated decay rates.

// todo decay overview figure

Also the application of additional data augmentation was not successfully. Instead the model was even unable to improve the training loss for some setups. Besides that the dropout layer had nearly no effect. Due to the usage of a pre-trained model *Dropout* could only be used easily within the custom layers forming the classifier. Applying *Dropout* to other hidden layers of the pre-trained model requires a more complex change to the predefined model. Thus we decided to not try this out. As alternative to the mentioned regularization methods we disabled the weight updates for all layers within the pre-trained model, namely the feature extraction module. Performing updates for the classification layer only and reducing

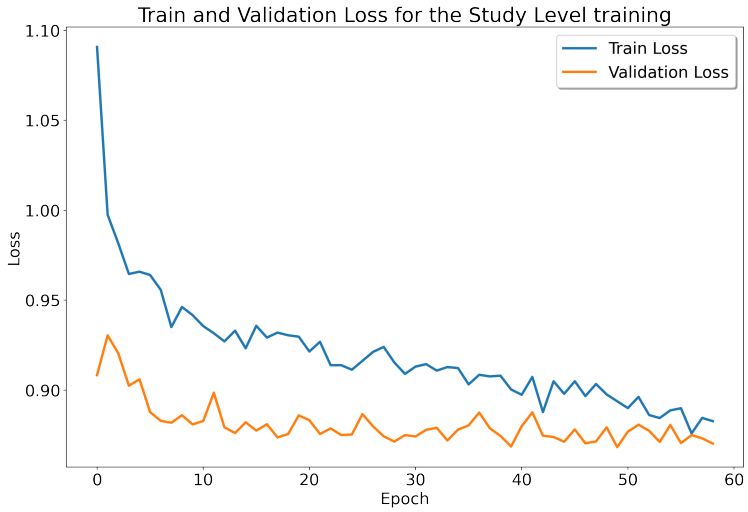

Figure 12: Training and validation loss for final training of the study level model.

the amount of trainable parameters stopped the overfitting of the model successfully. Still we had some problems with fluctuating validation losses or even increasing training losses after a first decrease. Thus we performed a manual hyper-parameter search. This was done by altering the training script to read in all parameters from a configuration file and creating a couple of reasonable combinations. We then executed all configurations within a script and compared results later on.

As result we came up with a training and validation loss as shown in figure 12. Still especially the validation loss is highly fluctuating and the overall decrease of the loss is relative small. However compared to other validation losses reported for the hyper-parameter search this configuration reached the minimal validation loss. The training is executed for 58 epochs with a batch size of 15 and a learning rate $l_r = 0.0005$.

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 13 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 7) 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.

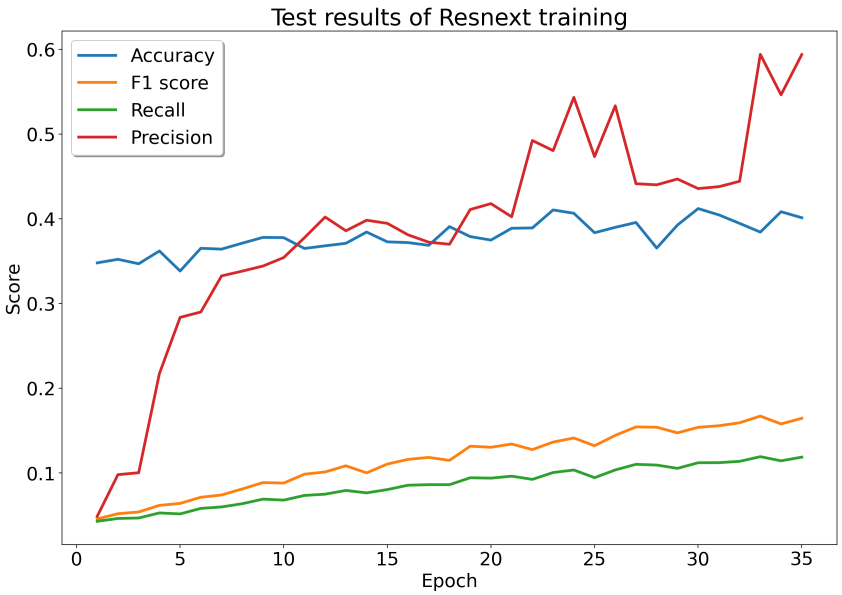

Figure 13: ResNeXt model metrics during training

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

To evaluate 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 an object detection model. However, we decided to utilize the widely used mean Average Precision (mAP) as our crucial performance evaluation metric since many of the contributions in the field of object detection adapted the metric making it the de-facto standard in many open object detection challenges like Pascal VOC [12] or COCO [34]. The mAP considers both, the quality of the class as well as 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 $< t$)[2]. There are various values for t conceivable depending also on the requirements 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 counted as a 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 [41] the authors propose to use an 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 reduced to 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 14 shows the validation metrics during pre-training on the RSNA data. The graph shows that during the first epochs of the pre-training 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 pre-training, rather that we experience a continuous learning for "introducing" the model and its weights to the problem domain.

[2]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.

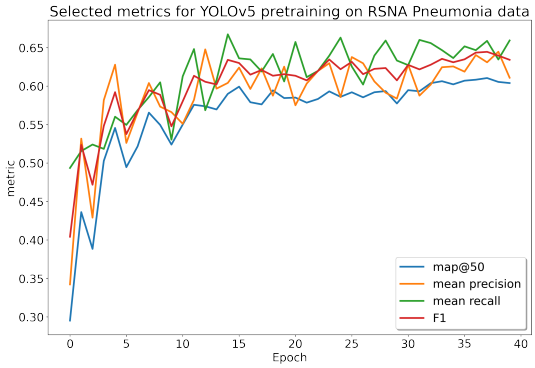

Figure 14: Validation results for the YOLO pretraining.

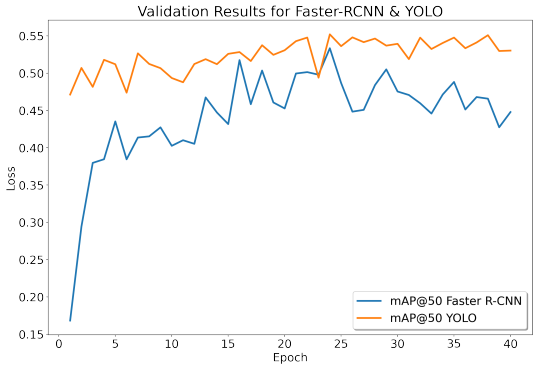

Figure 15: Validation results on the SIIM COVID-19 data.

The final validation results during training for both detection models on the COVID-19 dataset can be found in 15. The figure shows that we only see a slow increase in mAP@.50 for the YOLO model which is mainly caused by the pre-training. In contrast, the Faster R-CNN model shows a similar pattern like the YOLO in pre-training where with an 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 an 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 too long training. Due to time constraints we did not analyze this behavior in more detail and leave this open for further research, including hyper-parameter tuning and cross-validation methods. However, we chose 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 set consisting of 859 examples.

	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.

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 led us to expect. However, this reflects pretty much the model performances obtained in other research, where the Faster R-CNN on average always performs better than YOLO, where the advantage of 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 except the mean Precision, where the Faster R-CNN performs slightly better.

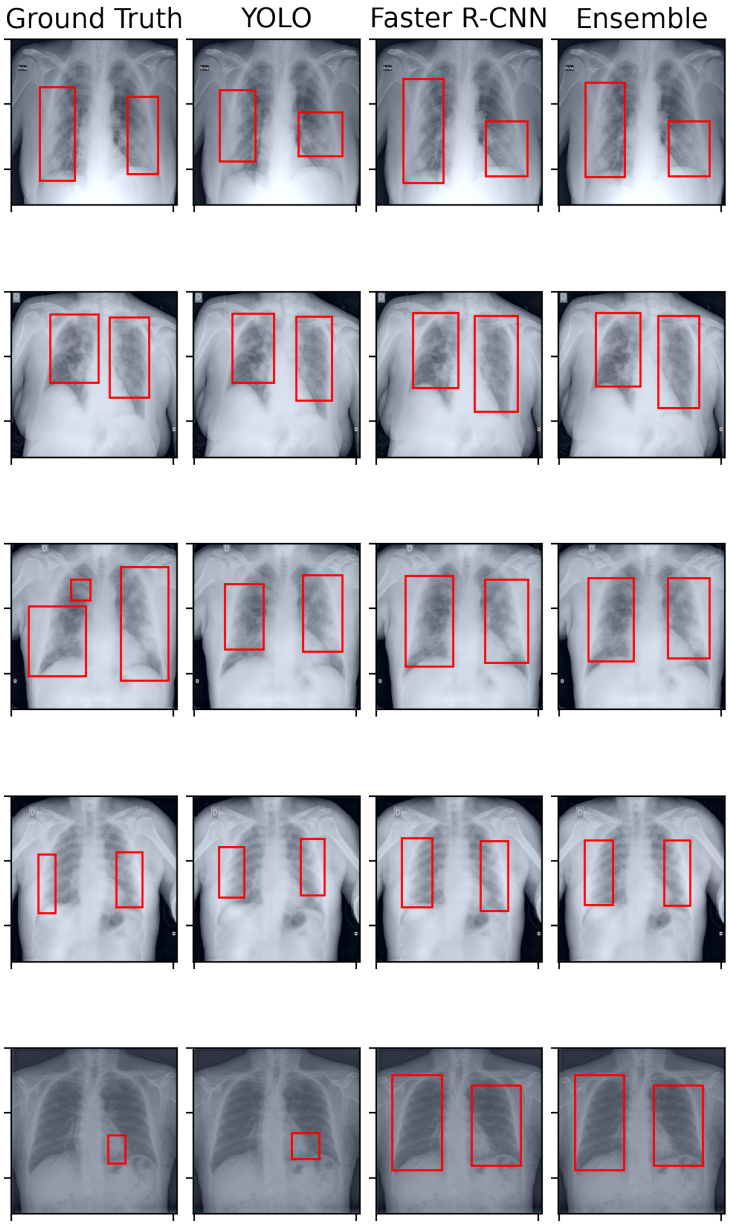

Figure 16: Prediction Results on random examples from the held-out set.

To show also a qualitative comparison, figure 16 shows five different samples from the held-

out set and their corresponding predictions of the Faster R-CNN, YOLO and Ensemble model. It can be observed that both the models already detect affected regions very well, especially in terms of absolute position and agree in most of the predictions, but also struggle in finding the perfect magnitudes of the boxes. Further, the results reveal also some problems of the models as visualized in the third and fifth example, where both models and ultimately the ensemble fails to detect rather small bounding boxes. In addition, the YOLO predicts smaller boxes relative to the Faster R-CNN in most cases leading to final results that are mainly driven by the latter predictions despite us designing the weights such that both models should contribute equally which is shown in the fifth example of figure 16.

To conclude this section, we showed that both models perform well on the task of detecting COVID-19 affected regions on chest X-rays, reaching the best results by combining a YOLO and Faster R-CNN model by a confidence score dependent bounding box fusion, but still leaving some open room for improvements (as will be discussed in chapter 6).

4.3 Evaluation of Study-Level Model

Written by Torben Krieger

The training of the image classification model was not straight forward and even after various optimizations the validation loss fluctuated. Figure 17 shows Accuracy, Precision, Recall and F1 score for the performed validations during training. Although all metrics are increasing over the training time, the slope of the increase is small. The total difference between the starting values and the metrics after 58 epochs is limited. The Precision metric shows most fluctuations, one of the highest values was already reached within the second epoch. Only within the last quarter of the training the fluctuation seems to be lower. One reason for less fluctuation at the end of the training could be the used *Cosine Annealing*, therefore the smaller learning rate prevents to big updates.

After the training a test of the image classification model was executed using a test set. The final metrics for that test could be found in table 2. Although the metrics are better as the results for the initial pre-training on the NIH dataset we have to stress here that the classification was done for four classes only. The metrics include also the mAP which is quite uncommon for classification tasks. Reasons to include it here is that the submissions of the Kaggle Challenge are rated using mAP@50 for both, the detection and classification task, in a combined fashion. The formal submission file has to provide a pre-defined dummy bounding box for each study-level classification. The IoU threshold of 50% is therefore irrelevant for the study task and the metric simplifies to simple mAP. The calculation of the mAP value for our test allows us to compare our results with competitors of the challenge without doing a formal submission. As reference, the first place of the challenge reported a mAP of 0.598 for the classification task only [11]. However they are using an ensemble of models with greater input sizes and performed pre-training on several other datasets as well

Figure 17: Validation metrics of the classification model during training.

	Study-Level Model
Accuracy	**0.669**
mean Recall	**0.495**
mean Precision	**0.541**
F1-score	**0.482**
mAP	**0.405**

Table 2: Test metrics for the image classification model.

as specifically for detection of normal pneumonia first. Still it has to be noted that the mAP values reported by the competitors were calculated on the hidden held-out dataset of the challenge. Figure 18 contains the confusion matrix for the test run. It is observable that our model classifies most images as *Typical*. Besides this the class *Negative* is classified often. The two remaining classes *Indeterminate* or *Atypical* are predicted seldom. By comparing with section 2.3 it is detectable that this roughly maps to the imbalance of the SIIM dataset. Thus over sampling the dataset or weighting of the loss function for underrepresented classes would be the next attempts to improve the classification. However we did not had the time to test these approaches. Apart from this the confusion matrix indicates that most images of the classes *Typical* and *Negative* were predicted correctly. Closing this section we have to conclude that our image classification model not meets the expectations we had. However at least for the two most important classes out of the four classes a acceptable classification performance is observable. Which means that our model is at least somehow able to tackle the underlying problem to detect COVID-19. Possible improvements or alternatives will be discussed in chapter 6.

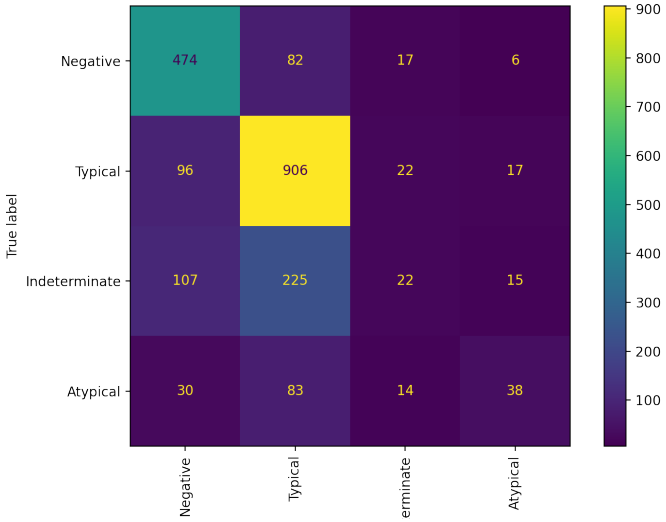

Figure 18: Confusion matrix for the image level classification.

5 Proposed web application

Written by Tobias Richstein

Since we do not plan to hand in a sample notebook to the Kaeggle challenge, we decided to build some sort of application that makes the result of our work a little more accessible and easily usable. In the end we settled on providing access to the models via a website that runs inference on the backend. We settled on using the framework *Flask* [53] since it uses Python just as the rest of the project.

The interface which can be seen in figure 19 on the following page is very simple but functional and built with the templating engine *Jinja 2* that is included with Flask. The user can first upload an image that is then previewed before choosing which model configuration to run: Faster R-CNN, YOLO or the ensemble of both. When clicking the submit button inference is run on the backend with the selected bounding box model as well as the study level model and the result is returned by re-rendering the webpage with the bounding boxes that were predicted. Their scores and coordinates also appear alongside the image. The output of the study level model is shown as well by the scores for the individual classes. The highest score is the one that the model has predicted.

We also provide a Dockerfile to containerize the entire application including the models to easily be able to build and run our results anywhere. Additionally a prebuilt Docker image is available on Docker Hub and can be run with: `docker run -p 5000:5000 tobiasrst/aml-project:latest`. The app will then be available on the host on port

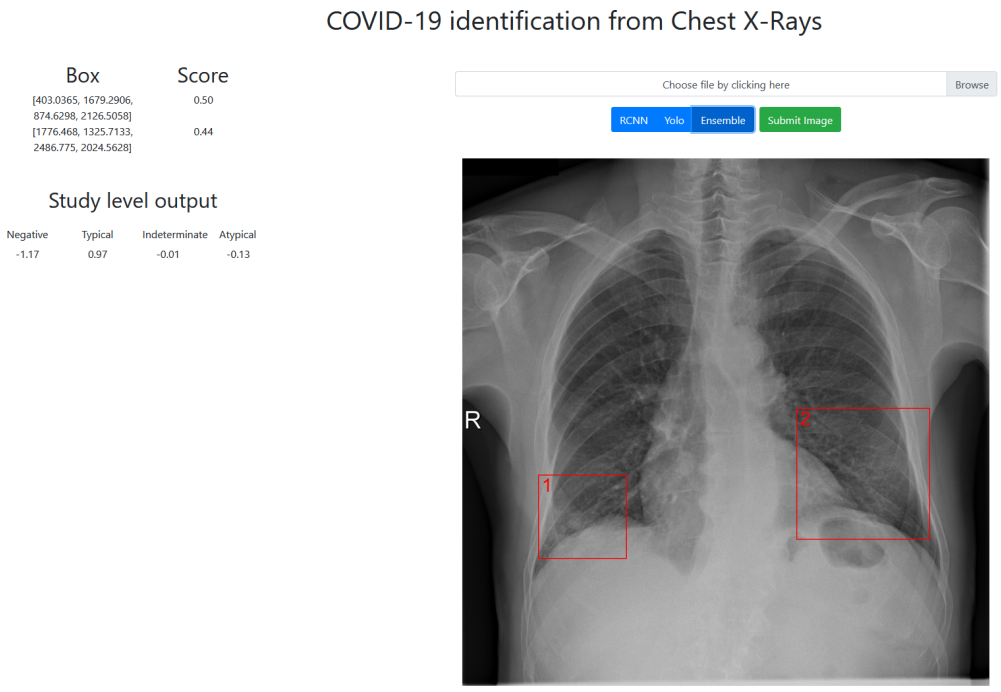

Figure 19: The web-app interface with a returned result

5000. Using the sample image located in the project directory at `/src/app/example.png` the inference can be tested quickly.

6 Conclusion

Written by All

In this project we wanted to create a CAD like system that is capable of detecting and classifying COVID-19 infections based on Chest X-Ray images by solving the eponymous Kaeggle challenge. Our solution consists of a multi-stage approach including several deep neural networks to achieve the goal set by the challenge creators. We showed that a YOLO and a Faster R-CNN are suitable candidates for detecting and localizing COVID-19 regions of interests. Since the dataset required by the challenge is limited, we decided to include the RSNA and NIH data from previous challenges to pre-train our model weights. We experienced that the Faster R-CNN outperforms the YOLO model on our test set, which is also consistent with other work in the field of object detection. However, for our final COVID-19 detection, we created an ensemble model that uses a weighted box fusion of both model outputs ultimately leading to the best results with a mAP@.5 score of 0.451. In general we are pleased with the final outcome of the COVID-19 detection performance and think that our work can be used as a solid basis for further research in this area. Due to the limited data we saw some overfitting effects with increased training time, which may be a starting point for further research. This may involve a repetition of our experiments with more data, an ablation study on how the pre-training of both models contribute to the performance or may be a root cause for overfitting effects, a hyper-parameter optimization process or even including more models with different architectures into the ensemble combination. Natural candidates would be for example a RetinaNet [32] or a segmentation-based mask R-CNN [21]. Another interesting extension may include research about an end-to-end train process of the ensemble approach, meaning that there could be an improvement in performance if both single models would be trained together in a joint fashion. However, we only had limited compute resources and time budget which is why we leave this open for further research.

TORBENS CONCLUSION HERE

// compare/test comparision to other diseases / data - how well we can distinguish

rescale images to 8 bit w.r.t. tissue/bones (..)

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 [70]. 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 [52]. 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.

All in all, we liked working on this project. We think that with the global situation around COVID-19, ML may be a key-factor in solving different issues regarding medical image analysis while still keeping the aforementioned concerns in mind. Overall we are very pleased with our results, also achieving all goals set by the underlying Kaeggle Challenge.

// todo relative for classification

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] 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.

[8] A. J. DeGrave, J. D. Janizek, and S.-I. Lee, "Ai for radiographic covid-19 detection selects shortcuts over signal," *Nature Machine Intelligence*, pp. 1–10, 2021.

[9] 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.

[10] "Digital Imaging and Communications in Medicine (DICOM)," National Electrical Manufacturers Association, Rosslyn, VA, USA, Standard NEMA PS3 / ISO 12052, PS 3.1-PS 3.12, 2018.

[11] N. B. Dung, "SIIM-FISABIO-RSNA COVID-19 Detection - 1st place solution," 2021. [Online]. Available: https://www.kaggle.com/c/siim-covid19-detection/discussion/263658.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[20] 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.

[21] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. arXiv: 1703.06870. [Online]. Available: http://arxiv.org/abs/1703.06870.

[22] 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.

[23] 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.

[24] 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.

[25] A. Jacobi, M. Chung, A. Bernheim, and C. Eber, "Portable chest x-ray in coronavirus disease-19 (covid-19): A pictorial review," *Clinical imaging*, vol. 64, pp. 35–42, 2020.

[26] 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.

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

[28] Kaggle, "SIIM-FISABIO-RSNA COVID-19 Detection - Identify and localize COVID-19 abnormalities on chest radiographs," 2021. [Online]. Available: https://www.kaggle.com/c/siim-covid19-detection/.

[29] 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.

[30] 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).

[31] P. Lakhani, "SIIM-FISABIO-RSNA COVID-19 Detection - Annotation Grading Methodology," 2021. [Online]. Available: https://www.kaggle.com/c/siim-covid19-detection/discussion/240250.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *arXiv:1708.02002 [cs]*, Feb. 7, 2018. arXiv: 1708.02002.

[33] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common objects in context," *arXiv:1405.0312 [cs]*, Feb. 20, 2015. arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

[34] 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.

[35] D. E. Litmanovich, M. Chung, R. R. Kirkbride, G. Kicska, and J. P. Kanne, "Review of chest radiograph findings of covid-19 pneumonia and suggested reporting language," *Journal of thoracic imaging*, vol. 35, no. 6, pp. 354–360, 2020.

[36] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," *arXiv:1512.02325 [cs]*, vol. 9905, pp. 21–37, 2016. DOI: 10.1007/978-3-319-46448-0_2. arXiv: 1512.02325.

[37] 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.

[38] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[39] 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].

[40] 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.

[41] 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.

[42] 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.

[43] S. Podder, S. Bhattacharjee, A. Roy, and Department of Electronics, West Bengal State University, Barasat, Kolkata, India 700126, "An efficient method of detection of COVID-19 using mask r-CNN on chest x-ray images," *AIMS Biophysics*, vol. 8, no. 3, pp. 281–290, 2021, ISSN: 2377-9098. DOI: 10.3934/biophy.2021022. [Online]. Available: http://www.aimspress.com/article/doi/10.3934/biophy.2021022 (visited on 09/18/2021).

[44] 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.

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

[46] 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].

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

[52] 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).

[53] A. Ronacher, *Flask: Web development, one drop at a time*. [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/ (visited on 09/17/2021).

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

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] E. B. Tsai, S. Simpson, M. P. Lungren, M. Hershman, L. Roshkovan, E. Colak, B. J. Erickson, G. Shih, A. Stein, J. Kalpathy-Cramer, *et al.*, "The rsna international covid-19 open radiology database (ricord)," *Radiology*, vol. 299, no. 1, E204–E213, 2021.

[60] 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.

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

[62] 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.

[63] M. d. l. I. Vaya, J. M. Saborit, J. A. Montell, A. Pertusa, A. Bustos, M. Cazorla, J. Galant, X. Barber, D. Orozco-Beltran, F. Garcia-Garcia, *et al.*, "Bimcv covid-19+: A large annotated dataset of rx and ct images from covid-19 patients," *arXiv preprint arXiv:2006.01174*, 2020.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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).

[71] 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.

[72] 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.

[73] 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.

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

[75] 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.

[76] 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.