UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Ruprecht-Karls-Universität
Heidelberg

Final project report

Advanced Machine Learning

Prof. Dr. Köthe

SIIM-FISABIO-RSNA COVID-19 Detection

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

Contents

1 Introduction and problem definition

1.1 Kaeggle Challenge

Written by Torben

1.2 Related Work

Written by Julian

COVIDNET beste [6]

1.3 Proposed Solution

Written by Tobias

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

2 Data

2.1 NIH Data

Written by Tobias

2.2 RSNA Data

Written by ?

2.3 SIIM COVID-19 Data

Written by ?

3 COVID-19 Detection

3.1 Faster R-CNN

Written by Tobias Richstein

The Faster R-CNN network architecture proposed in [5] by Ren, He, Girshick, *et al.* is an evolutionary step in a line of Region Based CNNs (R-CNNs) which are Convolutional Neural Networks (CNNs) that can perform object detection on images. When given an image, an R-CNN is able to predict bounding boxes for detected objects and also classify them. Each predicted bounding box is also given a confidence score that expresses how reliable the model assumes this result is. State of the art Faster R-CNNs achieve mean Average Precision (mAP) scores with an Intersection over Union (IoU) threshold of 0.5 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.

Beschreibung Faster RCNN

Beschreibung Resnext

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 [4], which we did. This reference implementation has also been pre-trained on the ImageNet dataset, meaning that we only fine-tune the weights to our use-case. We train the model on the NIH dataset described in section 2.1 on the previous page and only expect it to predict the classes of illnesses that can be seen in the X-rays. We encode the ground truths, consisting of the 14 classes of the NIH dataset, as one-hot vectors and therefore also expect output tensors of the same dimension. Like in the original ResNeXt paper, we also use a Stochastic Gradient Descent (SGD) optimizer that has Nesterov acceleration during training. Our learning rate decays over time and follows the equation given below which was originally proposed in [1] and modified slightly to provide a learning rate floor of 0.05:

$$\frac{1}{2}\left(1+\cos\left(\frac{t*\pi}{T}\right)\right)*0.95+0.05$$

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 to have a mean of $R = 0.485; G = 0.456; B = 0.406$ and a standard deviation of $R = 0.229; G = 0.224; B = 0.225$

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

Since we have somewhat limited hardware resources at our disposal in comparison to large scale compute clusters that are often used for such training tasks by researchers, we also apply a method called *Autocasting* to speed up training and allow us to use larger batch sizes. The basis of Autocasting is the ability to use mixed precision during network training. While most frameworks such as PyTorch usually use 32bit floating point numbers (single precision) for all calculations, it has been shown that performing some operations with 16bit representations (half precision) does not penalize accuracy but provides a large speedup since more data can fit in the most often constrained GPU memory and the also constrained data transfer bandwidth can be used more effectively [2]. 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 [3]. The loss numbers for the training and validation loss can be seen in 1 on the following page. It can be seen that in the end some overfit occures where the train loss keeps decreasing and the validation loss stays mostly constant or even increases very slightly. In the end we still decided to use the model after 35 epochs since the loss figures are very good and it also evaluates very well as will be shown later in chapter 4.1 on page 7.

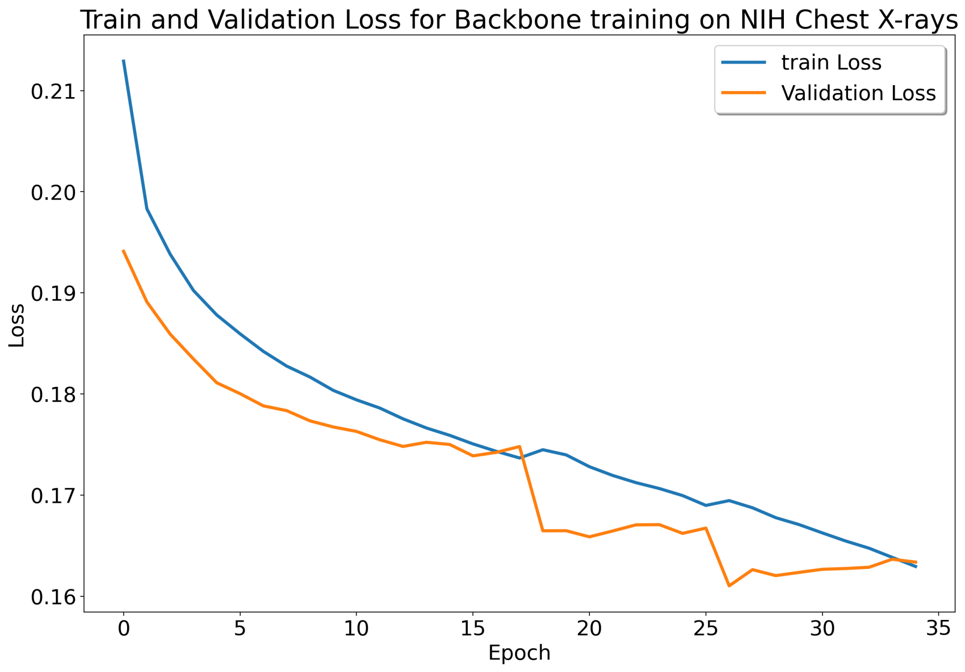

Figure 1: Loss figures of the ResNeXt training

Training of the Faster R-CNN

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

Due to the limited number of samples available in the SIIM dataset, we now augment the images more extensively to further prevent overfitting. Because we now have bounding boxes in the aforementioned (?) COCO format we also have to apply all augmentations to those too. To also allow the network to better detect small opacities and details we now train with 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

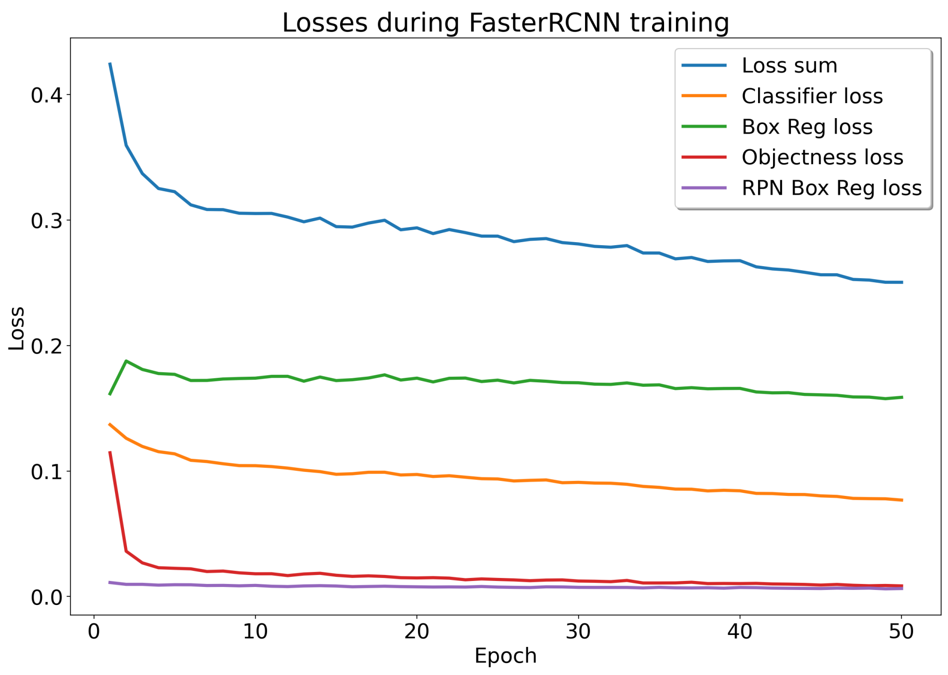

Figure 2: Loss figures of the Faster R-CNN training

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

3.2 YOLO

Written by Julian

3.3 Study-Level model

Written by Torben

4 Evaluation

4.1 Evaluation of ResNeXt

Written by Tobias

accuracy, f1, usw.

4.2 Evaluation of Faster R-CNN and YOLO

Written by Julian

4.3 Evaluation of Study-Level Model

Written by Torben

5 Proposed web application

Written by Tobias !

6 Conclusion

Written by All

References

[1] 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. 2018, arXiv: 1812.01187. [Online]. Available: `http://arxiv.org/abs/1812.01187` (visited on 09/07/2021).

[2] 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. 2018, arXiv: 1710.03740. [Online]. Available: `http://arxiv.org/abs/1710.03740` (visited on 09/07/2021).

[3] 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`.

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

[5] 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. 2016, arXiv: 1506.01497. [Online]. Available: `http://arxiv.org/abs/1506.01497` (visited on 09/06/2021).

[6] 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.

[7] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," *arXiv:1611.05431 [cs]*, Apr. 2017, arXiv: 1611.05431. [Online]. Available: `http://arxiv.org/abs/1611.05431` (visited on 09/07/2021).