Computer Vision Project Report - License Plate Detection and Recognition

Computer Vision - COME0824-001

Timur Carstensen, Max Keller, Lukas Kirchdorfer,
Tobias Sesterhenn, and Luisa Theobald

Faculty of Business Informatics and Mathematics
University of Mannheim, 68159 Mannheim, Germany
{luitheob, tcarsten, tsesterh, lkirchdo}@mail.uni-mannheim.de
max.keller@uni-ulm.de

Abstract. In this report, we investigate the use of various neural network architectures for license plate (LP) detection and recognition using the CCPD dataset. We first reproduce the results of an existing LP detection model, called RPnet, which is an end-to-end system that detects the bounding box of a LP in an image and recognizes the characters of the LP. We then explore the use of alternative neural network architectures, such as a Vision Transformer and a ResNet50 model, for LP recognition. In addition, we also try replacing the detection module of the RPnet model with a Vision Transformer to predict the LP characters. The results of these different approaches are evaluated and discussed in the report. Our best performing model uses a detection module with a ResNet50 as a backbone for the LP detection and a scaled bounding box prediction loss in the recognition module to accelerate learning. We match the performance of RPnet using fewer computational resources.

Keywords: License plate detection, Object detection, Object recognition, Object Segmentation, Convolutional neural network, Vision transformer

1 Introduction

License plate (LP) detection is one popular research topic in the field of Computer Vision. Due to that popularity, there exist various LP datasets [1][2][3][4] which can be used to train and evaluate different machine learning models. In this report we make use of the CCPD dataset by Xu et al. [4]. The authors propose an end-to-end license plate detection and recognition model, which in a first step detects the bounding box of a license plate in an image and then uses this information to recognize the respective characters of the license plate. The proposed RPnet model outperforms other license plate detection models on the CCPD dataset. Therefore, this model serves as a baseline to which we compare our own approaches to LP recognition models. In a first step we try to

reproduce the reported results in terms of bounding box detection and character recognition accuracy.

In a second step we replace the given model by different neural network architectures. One possibility is to use a Vision Transformer (ViT) [5] for the recognition of characters on the license plate. Although Transformers are commonly used in natural language processing, Dosovitskiy et al. [5] propose an adapted Vision Transformer for computer vision tasks that can potentially outperform the state of the art in image classification. Furthermore, we exchange the recognition module by Xu et al. [4] with a ResNet50 model [6] and compare their performance. In a further approach we also exchange the detection module by Xu et al. [4] with a Vision Transformer to predict the license plate characters. Finally, we present our best model: an end-to-end model with a ResNet50 backbone and a custom weighted loss function to accelerate the learning of the bounding box regression.[1]

The paper is structured as follows: after introducing the topic, some important concepts that were used in the project are explained in Section 2. Afterwards, the replication of the related paper's work is described in more detail in Section 3 before getting into the details with the conducted approaches in Section 4. Finally, the results are evaluated and discussed in Section 5.

2 Approach Overview

In this section, a short introduction into the modulation techniques used in the experiments is given. However, the following explanations will not give an exhaustive overview over all methods that will be presented in this paper. The interested reader is referred to the respective papers for further information.

2.1 Convolutional Neural Network (CNN)

As in O'Shea and Nash [7], CNNs are a type of artificial neural network that are designed to process data with a grid-like structure, such as images. They are composed of multiple layers, including convolutional layers and fully connected (or dense) layers. Convolutional layers apply filters to the input data to extract features such as edges, corners, and patterns, while fully connected layers combine and integrate this information to make predictions or classifications. CNNs are able to learn and extract features directly from the data and can be trained on large datasets, but they can also require a large amount of computational resources and may be prone to overfitting. They are widely used in a variety of applications, including image classification, object detection, and natural language processing.

[1] The source code for all our models is provided in the following publicly accessible GitHub Repository: github.com/timurcarstensen/come-0824-computer-vision-project

2.2 ResNet

Adding more hidden layers to the neural network helps to get higher level feature representations from the input image. However, training deep networks poses multiple problems such as vanishing and exploding gradients. To enhance the training of deep neural networks, He et al. [6] introduced the idea of residual learning which includes skip connections to avoid the vanishing gradient problem. The resulting ResNet50 model, which can be seen in Figure 1, is widely used in computer vision tasks. As it is pretrained[2] on a wide variety of images, we evaluate its usage in the detection of the LP bounding boxes as well as providing useful feature representations for the downstream recognition task.

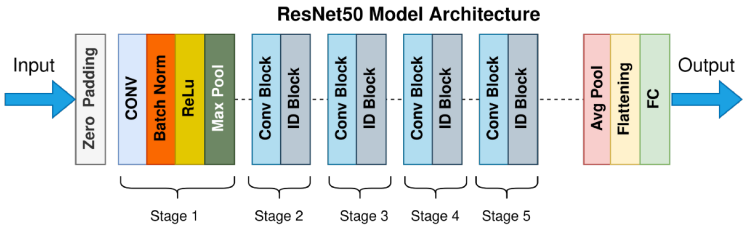

Fig. 1. The ResNet50 model architecture according to He et al. [6]

2.3 Transformer

Transformers are widely used in Natural Language processing (NLP) as described in Dosovitskiy et al.[5]. Usually they consist of an encoder block and a decoder block as described in Vaswani et al. [9], which is depicted in Figure 2. The architecture allows to effectively build translators for different languages. To give a coarse overview over how they work and how the concept of attention is used, a quick overview on transformers in the context of NLP is given. A sentence consists of multiple words where the connection of these words constitute the semantics of the sentence. This fact is used for the attention mechanism. The transformer encoder block uses the information provided by the sequence to learn contextualized input representations. With the semantics being the same in multiple languages, the decoder block must then form a valid sentence given the contextualized input representations provided by the encoder. In the encoder and decoder block of the transformer we attend over the same input sequence multiple times using multiple attention "heads". Each head is able to attend to/over different parts of the input and learn different representations. This is what is commonly called multi-head attention.

[2] In particular, we use the weights of a ResNet50 that was pretrained on the ImageNet dataset [8].

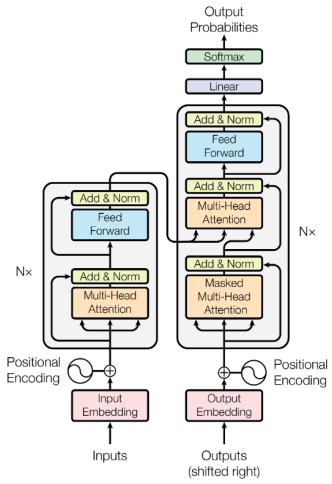

Fig. 2. The transformer encoder decoder architecture proposed by Vaswani et al. [9]

One approach to use a transformer for computer vision tasks is proposed by Dosovitskiy et al. [5] with the concept of Vision Transformers (ViT). The architecture is depicted in Figure 3 To create a sequence, the image is first divided into patches of a fixed size. After a flattening transformation the patches are then fed together with a positional embedding into a transformer encoder. Finally, using a MLP head, the architecture is able to classify the input image into distinct classes. We want to make use of this approach in our project, by applying it for the classification of the license plate characters. There is also already an existing approach by Moussa et al. [10] using a Vision Transformer for Forensic License Plate Recognition. The main benefit of the Vision Transformer in this case is that it is able to restore license plate characters from blurred license plate images.

2.4 Related work

The main contributions of the original paper by Xu et al. [4] are the large and publicly available dataset to train the network on and the approach of splitting up the process of LP recognition into bounding box prediction (i.e. detection) and the subsequent recognition of the characters. To further enhance the stated accuracy of recognition we set out to re-implement the authors' model using the code provided in their public GitHub repository[3]. Thereafter, the idea is to perform experiments with different architectural modifications to improve upon the authors' results.

[3] CCPD GitHub repository, accessed 19.12.2022

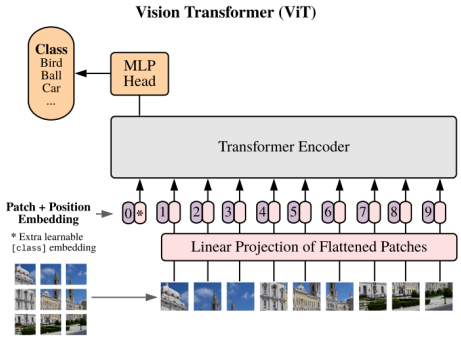

Fig. 3. The Vision Transformer architecture proposed by Dosovitskiy et al. [5]

3 Replication

This section will introduce the begin of the project work, starting with the re-implementation of the authors' model. Thereby, we need to get an understanding for the mostly undocumented GitHub repository which is based on outdated libraries, such as old versions of PyTorch that are neither usable or able to be installed at the time of writing. Since the given code was not executable, we rewrote the methods that implement the PyTorch dataset class[4] functionality and completly re-implemented the models using PyTorch Lightning. Here, we made use of the simplifications that PyTorch Lightning provides and integrated the training and validation steps into the class definitions of our models, thereby reducing complexity and improving the readability of our code.

Since, the dataset is both large by the numbers of samples (300.000 unique images) and by size the size of each sample (50 - 120 KB; ∼25.2 GB in total), we needed significant computational resources to train our models. For this reason we made use of the compute resources by the Data & Web Science Group at the University of Mannheim. Here we had access to the following machines:

- three machines with each 96 CPU cores, 1024 GB of RAM and eight RTX A6000 GPUs with 48 GB of VRAM
- three machines with 32 to 80 CPU cores, 768 GB of RAM and four RTX 2080 Ti GPUs with 11 GB of VRAM

The former were mainly used for training and the latter for pre-training due to the different number of parameters in each task. All machines were organized in a Storage Area Network (SAN) such that all files (except for temporary files) were shared across machines.

[4] PyTorch dataset template class, accessed 19.12.2022

3.1 Dataset

The license plate data set used consists of 250.000 unique license plate images and was compiled by Xu et al. [4]. Since the release of the original paper, the authors have updated and extended the dataset to include a total of 340.000 unique images. Each image has dimensions of 720 (width) by 1160 (height) by 3 (channels). The average size of each image file is around 200 kilobytes. The images were collected by a city parking management company in a provincial capital in China. With this, the dataset is the largest publicly available license plate dataset including several uncontrolled conditions such as rotation, challenging weather conditions such as snow or rain, distortion and vagueness as seen in 4. The annotation labels are detailed in table 3.1.

Fig. 4. License plate bounding boxes as in Xu et al. [4]

Label	**Description**
License Plate number	Each license plate number is comprised of a Chinese character (i.e. the province), a letter and five letters or numbers
License plate bounding box	This label contains the x and y coordinates of the top left and bottom right corner of the bounding box
Four vertices locations	This label contains the x and y coordinates of the four vertices of the license plate in the whole image and is used for segmenting the borders of the license plate
Horizontal and vertical tilt degree	the horizontal tilt degree is the angle between the license plate and the horizontal line. After the 2D rotation, the vertical tilt degree is the angle between the left border line of the license plate and the horizontal line

Table 1. Dataset annotations as in [4].

We split the base images into 100k examples for training and 100k examples for validation.

3.2 Replication Results

The architecture proposed by Xu et al. [4] can be seen in Figure 5. It is split into a detection module which predicts the bounding box of the LP and a recognition module which performs the LP character prediction. The first module consists of a set of convolution and pooling layers to finally perform the box regression. The recognition module uses the predicted bounding box coordinates to perform ROI pooling ([11]) over the feature representations of the input image obtained from the outputs of the different convolutional and pooling blocks from the detection module. These pooled representations are then fed to seven (7) classifiers to predict the individual LP characters. To improve the performance of the model, the authors pretrained the detection module on the training set for 300 epochs before then training the entire model (i.e. detection and recognition together) for another 100 epochs to reach their final results.

The authors used SGD with a learning rate of 0.001 and momentum of 0.9. They additionally decayed the learning rate by a factor of 10 every five (5) epochs.

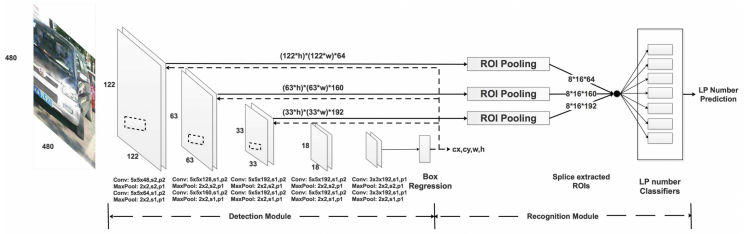

Fig. 5. Network architecture proposed by Xu et al. [4]

We considered three different approaches to replicate the results from the related paper:

- Pre-training the bounding box regression from scratch and using the weights to initialise the detection module.
- No pre-training before training the classifier (i.e. directly training the entire model end-to-end)
- Using the author's pre-trained weights to initialise the detection module.

As shown in Figure 6, we were not able to reproduce the authors' results by either pre-training from scratch or training the model directly end-to-end. Only when we used the pre-trained weights by the authors for the detection module were we able to achieve the recognition performance cited in the original paper. Notably, our models completely fail to reach the levels Intersection-over-Union (IoU) and generalized-IoU (gIoU) (cf. [12]) that are seen in the original paper. This is despite us using functionally the same models (only now they are

re-implemented using a more up-to-date PyTorch version) and the same hyperparamters. Two possible reasons are either: (a) errors in our implementation, or (b) the underlying PyTorch implementation changed/is inconsistent with the authors' implementation. For the latter, the authors implemented ROI pooling manually since PyTorch did not provide a built-in function at the time. We, on the other hand, use the built-in torchvision ROI pooling operation[5].

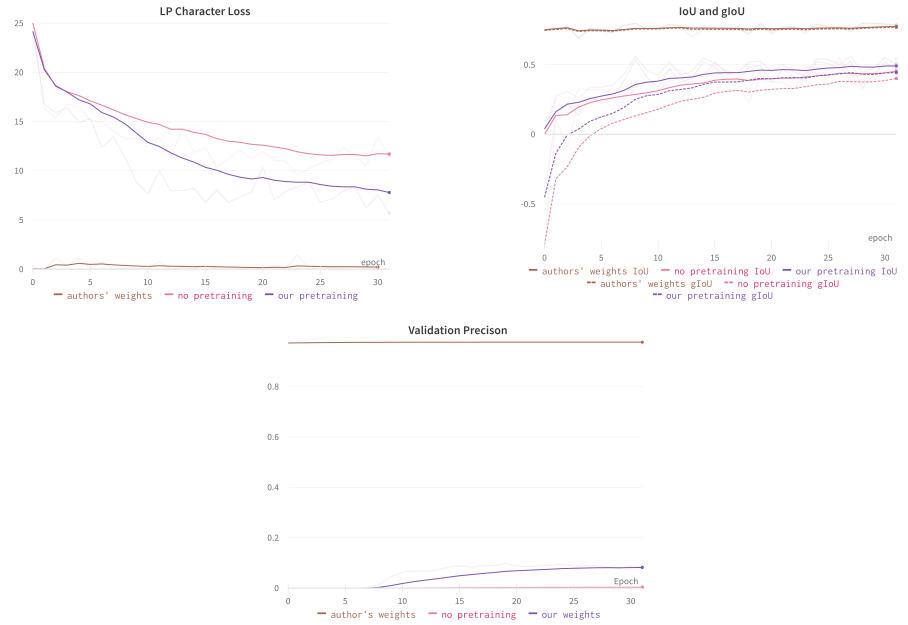

Fig. 6. Three graphs showing (from left to right, top to bottom) the learning curves for: (a) the LP character loss; (b) the IoU and gIoU, and (c) validation precision. Each plot shows the three configurations discussed in 3.2.

4 Methods and Experiments

The following sub-sections describe the custom approaches in more detail and evaluate their performance in comparison to the results from Xu et al. [4].

4.1 Approach 1: Minor adaptations

The first idea is to implement some minor changes into the proposed architecture. This includes to adapt the optimizer and the activation functions. Since the Adam optimizer is the most widely used optimizer for deep learning problems, we compare its performance to that of SGD, which was used in the original

[5] torchvision: ROI pooling, accessed 19.12.2022

framework. Additionally, it is surprising that the fully connected layers in the original architecture do not include any non-linear activation functions. Therefore, including ReLU's into the network to generate non-linearities is another interesting adaption.

The results of these minor changes as depicted in Figure 7 show that the Adam optimizer can be superior compared to SGD while non-linear activation functions have only a small effect on the performance.

The more interesting custom approaches such as ViT and ResNet are discussed in the following sections.

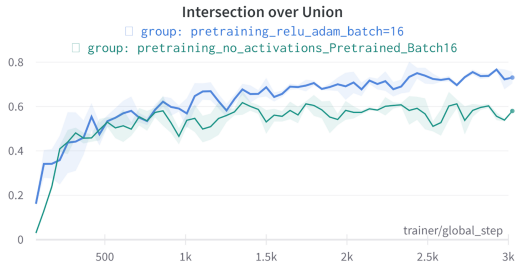

Fig. 7. The performance of using a ReLU and the Adam optimizer compared to the original implementation without ReLU and with SGD optimizer.

4.2 Approach 2: ViT for Detection and Recognition

Our first naive approach is to use a ViT for the LP recognition task in an end-to-end fashion. That is, there is no need to predict the bounding box anymore because the ViT is supposed to take an image and directly predict what characters the LP has. Therefore, the original image is divided into patches of size 32x32 which are then flattened and fed as a sequence into the transformer together with their respective positional embedding. The transformer consists of six layers using 16 heads, respectively.

Training was performed for 100 epochs using the Adam optimizer. The results are shown in Figure 8. We can clearly see that the classification performance of the model is very poor. The best accuracy after 100 epochs is 4% on the first classifier, which is only slightly better than random guessing (performance on train).

There could be serveral reasons why the training results are so poor. First of all, the architecture may still be too small to directly detect and classify license plates in given images. For example, as the input images have a dimensionality of 720x1160x3, respectively, and the license plate only covers a small subarea of the image, it may be hard for the model to detect the LP and classify the characters. Therefore, the architecture could be improved by increasing the depth

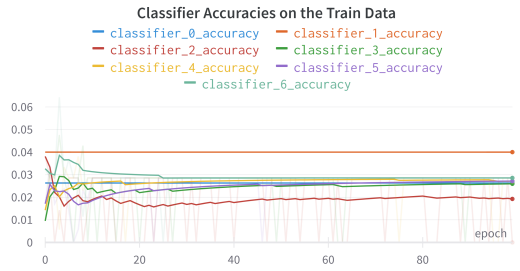

Fig. 8. The performance of using the End2End Vision Transformer with respect to the respective classifier accuracies.

and the number of heads. It may also help to increase or decrease the patch size of the architecture. Unfortunately, we did not have the resources to extend the architecture at this time.

Therefore, we follow a second approach, where we first limit the input size of the transformer by handing it only the slice of the image which includes the license plate. The ViT is supposed to work then similarly to the one presented by Moussa et al. [10]. The approach is discussed in section 4.4, where a detection module with a ResNet50 as the backbone predicts the bounding box and passes an accordingly sliced image of the LP to the ViT.

4.3 Approach 3: Detection Module with ResNet backbone and FCNN as Recognition Module

Instead of using a ViT we also wanted to see if we could match or even exceed the authors' performance by tuning the base model slightly. We first opted to replace the convolution and pooling blocks of the original model with a model from the ResNet architecture.

We first adapted the detection module of the original model and replaced the convolution and pooling blocks with either a: (a) ResNet50, and (b) ResNet152. We initialised both models with the above mentioned pre-trained weights and then pre-trained the "new" detection module on our bounding box prediction task. The Intersection-over-Union (IoU) and generalized-IoU (gIoU) and the bounding box prediction loss can be seen in 9.

Our reasoning for this is the following: (a) the ResNet architecture allows us to extract feature representations from the outputs of each residual block/layer ("bottleneck"). This allows us to perform ROI pooling over the feature representations as in the original model. (b) We can easily use ResNet models of varying depth to suit our needs and computational resources (18 to 152 layers)[6]. (c) Due to the skip connections, which are the defining feature of the ResNet architecture, we presume that we can learn suitable representations and the bounding

[6] torchvision ResNet model builder, accessed 19.12.2022

box regression more quickly. Finally, (d) we can make use of pre-trained weights for the ResNet where the model was trained on a classification task on the ImageNet dataset as discussed previously.

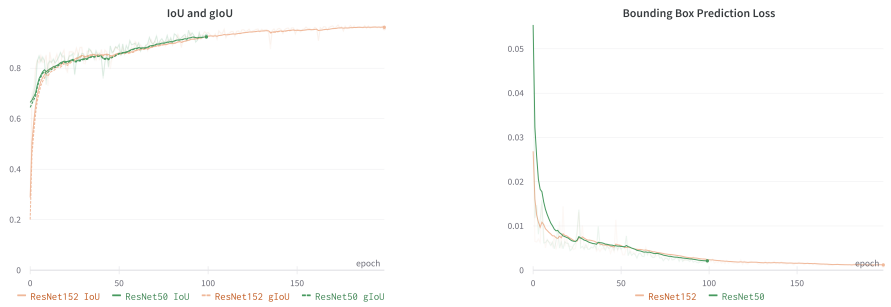

Fig. 9. For the detection module using either the ResNet50 or ResNet152: (a) IoU and gIoU, and (b) bounding box prediction loss. Both metrics are for the training set.

The results show that both models perform very well. In fact, the model using the ResNet50 is more or less indistinguishable from the model using the deeper ResNet152. At this point we must note that we only trained the ResNet50 for 100 epochs and the ResNet152 for 200 epochs. However, since the performance was so similar up until 100 epochs, we decided to use the ResNet50 based detection module going forward. This also lessens the computational requirements since ResNet50 only has ∼23 mil. parameters versus the ∼60 mil. of ResNet152.

During our previous attempts at replicating the authors' results we noticed that the magnitude of the bounding box prediction loss was two orders of magnitude smaller than the LP character classification loss. We therefore hypothesized that learning the correct bounding box prediction/regression in the original configuration was very slow since the low absolute value of the loss would also induce relatively smaller gradients of the loss w.r.t. to the parameters that are responsible for the bounding box prediction (i.e. model parameters in the ResNet backbone and the regression head). We thus conjectured that scaling the bounding box loss by some constant > 1 would speed-up the learning process of the bounding box. This should also improve the recognition task since we perform ROI pooling using the predicted bounding box. To put our hypothesis to the test we trained two architecturally identical models but where we scaled the bounding loss by a factor of 10 in the one models. That is, the total loss is now:

$$\text{loss} = 10 * \text{bounding\_loss} + \text{character\_loss} \tag{1}$$

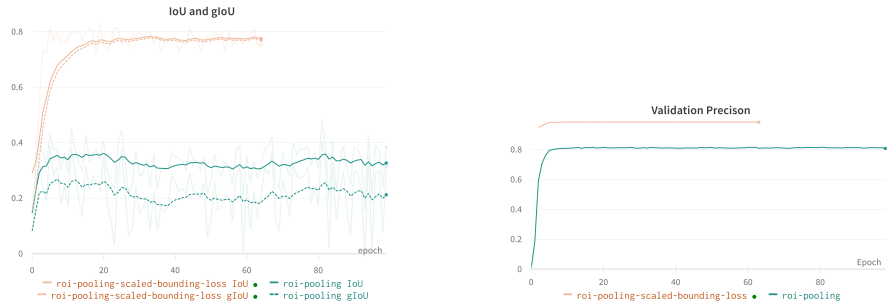

Fig. 10. For the entire model using the detection module that uses a ResNet50 as the backbone: (a) IoU and gIoU, and (b) validation set precision for the license plate character predictions.

As we can see in Figure 10, the model that used the scaled bounding box prediction loss not only learns the bounding box prediction much faster, but it continuously outperforms the model without the scaling factor. The same can be said for the validation precision. The model only needed 5 epochs to reach a validation precision of 0.9812 while the alternative never reached that performance level. We also wanted to see if changing the optimizer (Adam instead of SGD; not reported here) made a notable difference but observed that it performed worse than both models shown in the figure.

4.4 Approach 4: ResNet50 as Detection Module and Vision Transformer as Recognition Module

As described in Section 4.2 we want to evaluate the performance of a ViT as a recognition module with the pre-trained ResNet50 as a backbone architecture. This means that the detection module uses a ResNet50 to predict the bounding box of a license plate in a given image. The cropped image slice is then input into the ViT which performs the LP character prediction. The ViT has a patch size of 8x8, a depth of 6 and uses 16 heads, respectively.

The results of the training are depicted in Figure 11. It is clearly to see that the performance of the detection module with the ResNet50 on finding the correct bounding boxes is good, achieving an IoU of over 0.9 on the training set. Nevertheless, the performance of the ViT in the detection module for license plate characters is very poor, as the LP character loss converges after 50 epochs to a value of over 15. In detail, the model achieves an accuracy of over 90% on the first two classifiers and only 10% or less on each of the remaining classifiers.

Fig. 11. Three graphs showing (from left to right, top to bottom) the learning curves for: (a) the LP character loss; (b) the IoU and gIoU, and (c) validation precision. Each plot shows the configuration discussed in 4.4 with a ResNet50 as Detection Module and a Vision Transformer as Recognition Module.

4.5 Comparison of Results

As shown above, all models except for the one with the ResNet50 backbone and scaled bounding box loss fall far behind the results reported by the authors. Approaches 1, 2 and 4 fail to make any significant learning progress. In the best case the IoU/gIoU is learned quite well (cf. approach 4 4.4) while the LP classifiers converge at very high absolute losses. Across all unsuccessful approaches we do note however that the models manage to learn the prediction for the first and second LP character really well. That is, we achieve accuracies of ~ 0.98 in almost all cases for the first two characters. This is because of the homogeneity of the distribution for these particular characters. The first LP character is almost always the same since it designates the province and most pictures show LPs from cars in the same province. The same can be said for the 2nd character, which is "A" most of the time.

Our successful model showed similar performance as that of the authors. However, by employing a ResNet50 that was previously pre-trained on the ImageNet dataset we were able to significantly shorten pre-training and training time and thus save on computational resources. We only pre-trained our detection module with the pre-trained ResNet50 backbone for a total of 100 epochs (compared

with the authors' 300 epochs). Additionally, our model converged after around 10 epochs during the end-to-end training part (i.e. after initialising the detection module in recognition task with our pre-trained) weights. We credit this to our scaled bounding box prediction loss. We could see a dramatic improvement in convergence time when compared to the approach where we did not scale the loss.

5 Conclusion

In this paper, we presented different approaches for the task of LP detection and recognition. The work is based on the paper of Xu et al. [4] who propose an end-to-end architecture trained on the CCPD dataset. First of all, we reproduced the results from Xu et al. [4]. Afterwards, we developed different custom approaches to solve the same LP detection and recognition task and compared the respective performance in terms of IoU and accuracy. We can conclude that contrary to our expectations ViT's did not prove to be beneficial. One reason for their rather bad performance can be that ViT's might just not be suitable for the given problem. However, Moussa et al. [10] showed that they can work very well in the context of LP recognition. Another possible reason is that the chosen architecture of the ViT might be a poor choice for the given task. Therefore, having a deeper and broader ViT and choosing different patch sizes might improve the performance. However, due to limitied computational resources, this hypothesis could not be tested.

We showed that we can come up with a very competitive approach by using the ResNet as backbone model and scaling the bounding box prediction loss. The results are similar to those reported by Xu et al. [4], however we could reduce the computational complexity dramatically.

In future work, the proposed architectures can be tested in a more in depth manner, choosing different patch sizes and increasing the number of parameters of the Vision Transformer. Lastly, the CCPD test dataset should be expanded, as it currently does not provide the annotations to test on the IoU and gIoU.

6 Author Contributions

We would like to emphasize that the workload in this team project was equally distributed among all team members. Programming tasks were predominantly but not exclusively performed by Timur, Tobias and Lukas, whereas Max and Luisa were mainly responsible for research and idea creation. Therefore, we believe that we should all be graded equally. The chapters of the report were also distributed among the team members, with overlapping contributions. Here is a very rough overview about the responsibilities of everyone in the report, although it is not exhaustive:

Timur: Methods and Experiments (4.1, 4.3, 4.5), References
Max: Introduction, Approach Overview (2.1, 2.3), References
Lukas: Conclusion, Replication, References

Tobias: Introduction, Methods and Experiments (4.2, 4.4), References
Luisa: Approach Overview (2.2), Replication, References

7 References

[1] S. Azam and M. Islam, "Automatic license plate detection in hazardous condition," *Journal of Visual Communication and Image Representation*, vol. 36, Feb. 2016. DOI: 10.1016/j.jvcir.2016.01.015.

[2] G.-S. Hsu, A. Ambikapathi, S.-L. Chung, and C.-P. Su, "Robust license plate detection in the wild," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6. DOI: 10.1109/AVSS.2017.8078493.

[3] J. Špaňhel, J. Sochor, R. Juránek, A. Herout, L. Maršík, and P. Zemčík, "Holistic recognition of low quality license plates by cnn using track annotated data," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6. DOI: 10.1109/AVSS.2017.8078501.

[4] Z. Xu, W. Yang, A. Meng, *et al.*, "Towards end-to-end license plate detection and recognition: A large dataset and baseline," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Sep. 2018.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2020. DOI: 10.48550/ARXIV.2010.11929. [Online]. Available: https://arxiv.org/abs/2010.11929.

[6] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385. [Online]. Available: https://arxiv.org/abs/1512.03385.

[7] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *CoRR*, vol. abs/1511.08458, 2015. arXiv: 1511.08458. [Online]. Available: http://arxiv.org/abs/1511.08458.

[8] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[9] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010, ISBN: 9781510860964.

[10] D. Moussa, A. Maier, A. Spruck, J. Seiler, and C. Riess, *Forensic license plate recognition with compression-informed transformers*, 2022. DOI: 10.48550/ARXIV.2207.14686. [Online]. Available: https://arxiv.org/abs/2207.14686.

[11] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

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