UNIVERSITY OF BELGRADE
FACULTY OF MATHEMATICS

Nikola Grulović

TECHNIQUES OF DATA AUGMENTATION AND DATA CREATION FOR OBJECT DETECTION

Master thesis

Belgrade, 2022.

Mentor:

dr Milena Vujosevic Janičić, associate Professor
University of Belgrade, Faculty of Mathematics

Members of the commission:

dr Mladen Nikolić, associate Professor
University of Belgrade, Faculty of Mathematics

Lester de Abreu Faria, Professor at ITA & PHD in Microelectronics
IPFacens, Sorocaba, São Paulo, Brazil

Date of defense: _____

Master's thesis title: Techniques of data augmentation and data creation for object detection

Abstract: Automatic detection and recognition of objects has a very important role in everyday life. One of the most significant examples of the use of object detection and recognition is in autonomous driving. To enable efficient object recognition, machine learning methods are used, for example neural networks, deep learning and computer vision. However, to obtain results of the desired quality, a large amount of data, high processing power, and data labeling are usually required, which can be very expensive or largely unavailable. This paper deals with methods for expanding the data, as well as methods for reducing the time needed to train the model, in case a large amount of data or a large processing power is not available. In the paper, the data is expanded with standard augmentations and methods of creating new synthetic data. Using transfer learning reduces the time it takes neural networks to learn the main features of the available data. By using the mentioned methods, improved results are obtained compared to the current best model for object detection. Improvements to the model are reflected in better detection and better recognition of objects.

Keywords: artificial intelligence, machine learning, computer vision, deep learning, digital image, VGG16, YOLO, StyleGAN

Contents

Chapter 1

Introduction

Machine learning refers to the ability of a machine to draw conclusions using available data instead of strictly defined and coded rules. In fact, machine learning allows computers to learn by themselves. The results of machine learning indirectly occur in everyday life, from predicting the weather forecast to autonomous driving.

Supervised machine learning techniques involve the use of large amounts of labeled data. However, there are several reasons why large amounts of labeled data are not always available. Data collection and labeling is a process that can be extremely expensive or even impossible [4], such an example is, in the field of medicine, data labeling by radiology experts is expensive and not feasible on a large scale [25]. In such situations, machine learning has only a small amount of data at its disposal and it is necessary to apply special techniques and approaches in order to overcome the problem of availability of a small amount of data, if possible.

In this work, the problem of the availability of a small amount of data is solved, where the data are images with marked frames of objects. To solve the problem of a small amount of data, two approaches are used: expanding the dataset and transfer learning. The paper describes the process of expanding the dataset, whereby the data is expanded using standard techniques for image augmentation and data obtained using a model for generating synthetic images. Expanding the dataset using the above methods still does not solve the problem of small amount of data well enough. Transfer learning techniques [38] on an already trained model for object detection [5] are also described in the paper. Using transfer learning aims to learn representations from one domain and transfer the learned features to a closely related target domain [6, 3]. An indirect contribution of using transfer learning methods is the reduction of processing power required for model training. By using

the aforementioned methods of dataset expansion and transfer learning, a model with better characteristics is obtained.

In chapter 2, a brief overview of the object detection model used in this paper is given. Chapter 3 gives an overview of techniques and generative adversarial networks used in solving small data problems, while chapter 4 describes techniques for evaluating those models. In the head 5, the experiments and results of the models obtained by the mentioned techniques are presented. In chapter 6, the problems faced by this work, methods of overcoming them and possible further improvements of the solution are briefly presented.

Chapter 2

Object detection

Computer vision is a field of machine learning, which deals with the construction and use of systems for processing, analysis and interpretation of digital images or videos [2, 19]. Object detection is a computer vision technique whose task is to detect objects in an image. For successful object detection, it is necessary to train the model on a large amount of data. As it is sometimes impossible to obtain a sufficient amount of data, data augmentation is used: the data is expanded by adding modified or synthetically generated data [33].

2.1 The development of object detection

Object detection is used in a large number of applications, such as, for example, autonomous driving, robotic vision, video surveillance, etc. [39]. In the last two decades, progress in object detection has gone through two periods [39]: the "traditional object detection period" (before 2014 year), and the "deep learning-based object detection period" (after 2014).

The performance of object detection has been significantly improved thanks to deep convolutional neural networks (CNN), which has led to remarkable discoveries. The development of a deep convolutional neural network also enabled the detection of bounding boxes[1] and significant parts of the object in the images [12, 32, 21, 1] .

In the era of deep learning, object detection can be grouped into two types: *two-stage detection* and *one-stage detection* [39]. Within two-stage detection, the

[1]Bounding boxes represent coordinates on part or the whole image within which the recognized object is located.

first stage of the model is used to extract regions[2], while the second stage is used to classify and further refine objects. This method of object detection is relatively slow, but gives very precise results. One-stage object detection models refer to a class of models that skip the region proposal stage. Such models require only one pass through the neural network and predict all bounding boxes in one pass at the end. This type of model usually has faster detection, but is less accurate in detecting and recognizing objects.

2.2 Object detection model VGG16

The VGG16 model uses CNN and was considered one of the best computer vision models in 2014 after winning the ILSVR (Imagenet) contest [34]. The authors of this model looked at other models and noticed that by increasing the depth and using an architecture with smaller convolutional filters, performance can be improved. The change in architecture proved to be a significant improvement over previous models [34].

The VGG16 model belongs to the single-stage detection model and got its name from the sixteen layers that can learn, while the entire network consists of thirteen convolutional layers, five max pooling layers and three fully connected layers that are summed up in a total of twenty-one layers. The architecture of the VGG16 model is given in the figure 2.1.

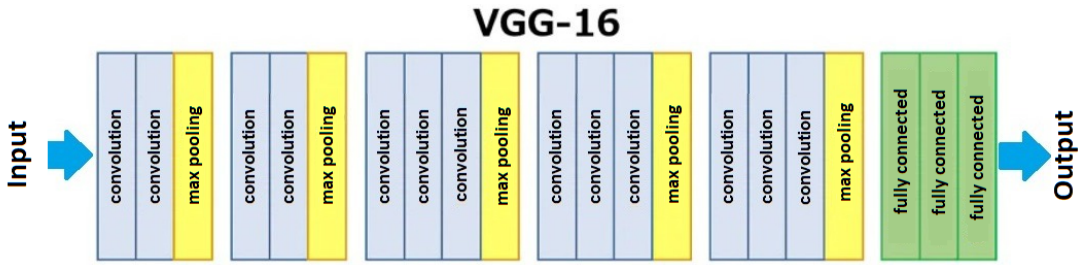

Figure 2.1: VGG16 architecture

The model was able to predict objects in the given validation images with 6.8% error outperforming the first subsequent model by 0.9%. [34]. The disadvantage of the VGG16 model is that it is not able to determine the frames of the detected object in the image.

[2]Regions represent parts of the image that may contain objects or parts of objects

2.3 Object detection model R-CNN

Due to the reuse of convolutional neural networks and the greater computing power available, in 2014 a model named regions with CNN features R-CNN appeared in 2014 [12]. The R-CNN model is the first two-stage detection model and consists of three modules that perform different tasks [12]:

1. The first module generates region suggestions using a selective search algorithm. The selective search algorithm performs image segmentation based on pixel intensity using a graph-based segmentation method [29]. Based on the obtained segmentation map, suggestions of regions are created that are used in the next module [35].

2. The second module is a large convolutional neural network. Each proposal is scaled to a fixed-size image and fed into a trained CNN model to extract a fixed-length feature vector.

3. The third module is a set of linear methods of support vector machines specific for each class, which recognize objects from the vector of characteristics.

The problems that arise with R-CNN are [10]:

- Due to the possible classification of a large number of region proposals per image, model training cannot be used in real time.

- The selective search algorithm is fixed and as such does not adapt to the data. This can result in poor region proposals being obtained in some situations.

During several iterations of the development of R-CNN, the model is significantly accelerated and new models are created: fast R-CNN [11] and faster R-CNN [32]. The fast R-CNN object detection model takes a similar approach to R-CNN with a few simple improvements to the model. The faster R-CNN object detection model is composed of two modules. The first module is a deep convolutional network that proposes regions, while the second module is an R-CNN model that uses proposed regions for object recognition. The entire system is a single, unified object detection network [32].

2.4 YOLO object detection model

The object detection model R-CNN uses regions to detect objects within an image. The model does not look at the complete image, but instead looks at parts of the image that have a high probability of containing objects. You only look once, YOLO for short, is an object detection model that differs from region-based models. The YOLO model belongs to the model based on one-stage detection and was created in 2015 [31]. With the YOLO model, a single convolutional network predicts regions and detects objects in bounding boxes.

Separate components of object detection are combined into one neural network. The model uses the found features from the whole image to predict each bounding box. Also, all bounding boxes for all objects in the image are predicted simultaneously. This means that the model thinks globally about the entire image and all the objects on it [31].

Chapter 3

Dataset Expansion Techniques

The dataset can be expanded by creating new data. Data creation techniques include data augmentation and generative adversarial networks. In some cases, even after expanding the training set, the amount of data available may be insufficient to train the model. Such a problem can be alleviated by the transfer learning technique.

3.1 The Small Data Set Problem

Models trained on a small dataset tend to find features that do not exist, resulting in high variance and high error on the validation set. In this work, two methods are used to avoid this behavior. The first method is to expand the dataset. The data is augmented with images that have been altered using standard augmentation methods and images that have been synthetically generated using generative adversarial networks. Another method is the use of transfer learning.

3.2 Standard augmentation methods

Standard augmentation methods include rotation, flip, scale, crop, blur and color augmentations. Randomly combining the mentioned methods creates new data that serves as an extension of the training dataset.

Flip

Image fliping refers to the creation of a new image that corresponds to the mirror image, with respect to the vertical or horizontal axis. It is not possible to apply

the flipping operation in every domain. For example, vertical flipping will not make much sense in the case of traffic vehicles. Examples of flipping are shown in the figure 3.1.

Figure 3.1: Flipping example: original image (left), vertical flipping (middle) and horizontal flipping (right) [28]

Rotation

Rotation is a transformation that creates new images that correspond to the original image rotated by an arbitrary angle. By applying the rotation operation, the dimensions of the image are often changed. If it is necessary to preserve the original dimensions, the image is cut as necessary, while the empty parts of the image are filled using a certain method. Examples of rotations are given in the figure 3.2.

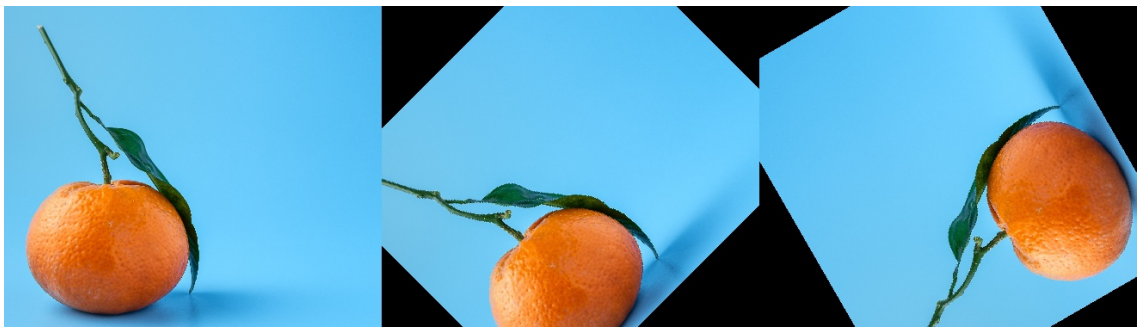

Figure 3.2: Rotation example: original image (left), 45° rotation (middle) and 120° rotation (right) [28]

Scaling

Scaling is a transformation that increases or decreases the image by a given factor that is greater than zero. When using a scaling factor greater than one, the di-

mensions of the final image are larger than the original. To preserve the original dimension, the new image is cropped to the original dimension. In case the scaling factor is between zero and one, the dimensions of the image are smaller than the original. In order to preserve the original dimensions, the empty space is filled using a certain method. Examples of scaling are given in figure 3.3.

Figure 3.3: Scaling example: original image (left), scale factor greater than 1 (middle) and scale factor between 0 and 1 (right) [28]

Cropping

Cropping an image refers to creating a new image by randomly sampling parts of the original image. The crop is then scaled to the dimensions of the original image. An example of cropping is given in the figure 3.4.

Figure 3.4: Crop example: original image (left), random crop (middle and right) [28]

Translation

Translation means moving the image in a horizontal or vertical direction. This method is very useful, because most objects can be located anywhere in the image.

The empty space that is created after the transfer is painted using appropriate methods. An example of translation is shown in the figure 3.5.

Figure 3.5: Translation example: original image (left), random translation (middle and right) [28]

Gaussian noise

Gaussian noise is the addition of random values from a normal distribution to the pixels of the original image. This creates a new image with noise. By adding Gaussian noise, all features in the data are deformed. This is important because there may be data in the training dataset that contain similar features, which can lead to overfitting of the model being trained on that set. An example of noise is given in the figure 3.6.

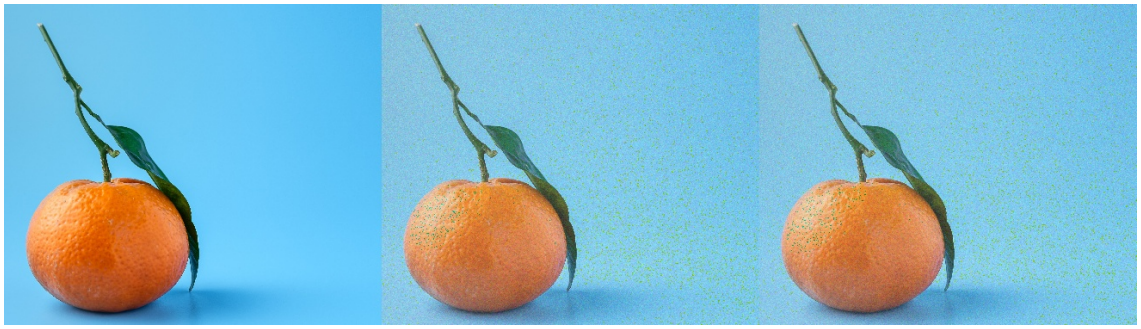

Figure 3.6: Example of adding noise: original image (left), mean 1 and standard deviation 0.1 (middle), mean 0 and standard deviation 0.5 (right) [28]

Gaussian blur

Gaussian image blur is an application of a convolution operation[1] with a Gaussian bell [27]. Increasing the standard deviation produces a stronger blurring effect. An example of blur is shown in the figure 3.7.

Figure 3.7: Blur example: original image (left), standard deviation 7 (middle), standard deviation 21 (right) [28]

Color Augmentation

Color augmentation changes the color properties of an image by changing the values of its pixels. Pixel values can be changed in different ways.

Brightness involves multiplying each pixel of the image by the corresponding illumination factor greater than zero. If the luminance factor is between zero and one, a darker image is obtained, and if it is greater than one, a brighter image is obtained. An example of brightness is shown in the figure 3.8.

Figure 3.8: Example of changing lighting, original image (left), lighting factor between 0 and 1 (middle), lighting factor greater than 1 (right) [28]

[1] convolution is an operation on two functions f and g that produces a third function $f * g$, which expresses how the shape of one is modified by the other

The technique of changing the contrast represents changing the degree of difference between the lighter and darker parts of the image. Increasing the contrast increases the difference between brighter and darker parts, while decreasing the contrast decreases the difference. An example of changing the contrast is shown in the figure 3.9.

Figure 3.9: Example of contrast change, original image (left), example of enhanced contrasts (middle and right) [28]

The technique of changing saturation of the image represents changes in the intensity of colors. By increasing the intensity, an image with more pronounced colors is obtained. An example of intensity change is shown in the figure 3.10.

Figure 3.10: Example of intensity change, original image (left), weaker increase in intensity (middle), stronger increase in intensity (right) [28]

The technique of changing the HUE moves the pixels in the image to another point on the color wheel. Random displacement generates new images with different colors. An example of a shade change is shown in figure 3.11.

Figure 3.11: Example of hue change, original image (left), shifted pixels for random values (middle and right) [28]

3.3 Generative adversarial networks

During the development of machine learning, the concept of generative adversarial networks was developed. In generative adversarial networks, there are two models: *generator* and *discriminator*. The generator is tasked with generating images and the discriminator is tasked with distinguishing generator-generated images from real images [13]. Generative adversarial networks rely heavily on large amounts of diverse and high-quality training data to generate photorealistic images [37]. They learn randomly and the result can be said to be good when the discriminator has an approximate accuracy of 50%, more precisely when it starts to guess whether the image is real or not. The basic architecture of generative adversarial networks is shown in figure 3.12.

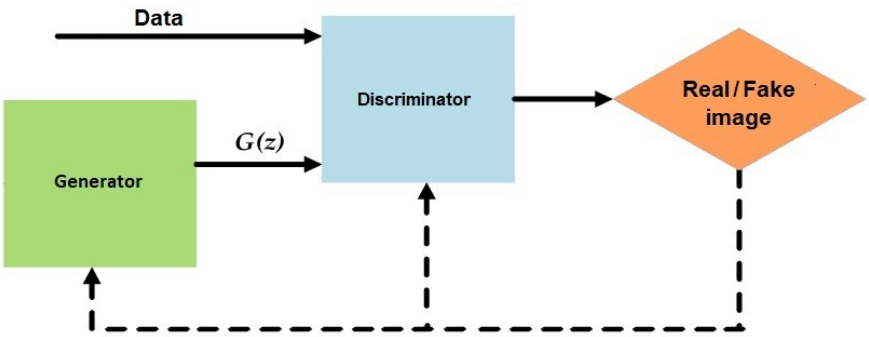

Figure 3.12: Generative adversarial networks: input data represent images, G(z) is the image generated by the generator [9]

Progressive generative adversarial networks modify the architecture in such a way that the training of generative adversarial networks starts with a low image resolution and then progressively increases the resolution by adding layers to the network. The training process is shown in the figure 3.13. This incremental nature allows the model during training to first detect the visual features that manifest on the initial layers, from standard features to fine details [22].

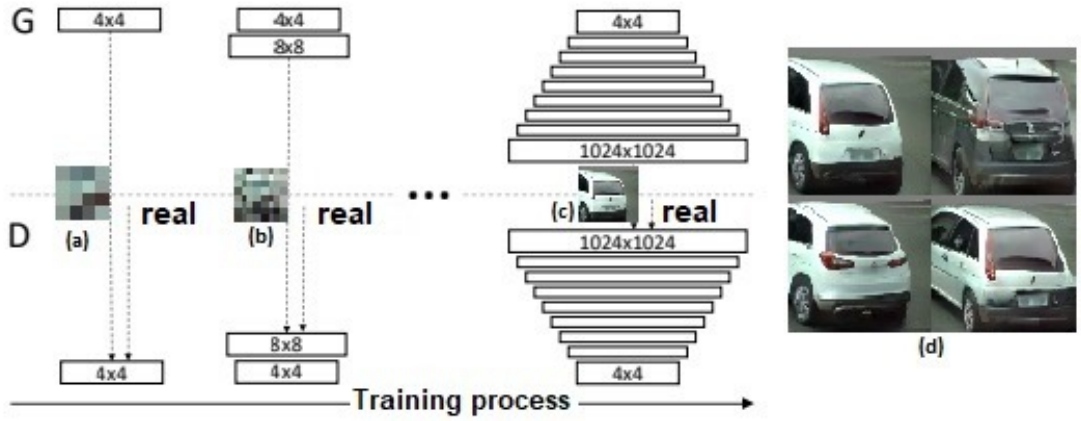

Figure 3.13: The training process of progressive generative adversarial networks starts from low resolution images (4×4) and progressively adds layers to the network and the resolution increases up to (1024×1024). The first image (image a) represents the first iteration of model training, the second image (image b) represents the second iteration of training, while the third image (image c) represents the last iteration. The car images (image d) represent the highest resolution generated images [22].

During the development of generative adversarial networks, most of the work was done on improving the discriminator, which led to better results. On the other hand, style generative adversarial network, or *StyleGAN* for short, is an addition to the architecture of generative adversarial networks, which introduces significant modifications to the generator model [23].

The style-based model generator consists of two parts and is shown in figure 3.14. The first part forms a fully connected network with eight layers. The network is represented as a function $f : Z \to W$ and is called a mapping network. The input of the network is the normalized latent code $z \in Z$. Latent code or z-vector z, is a vector containing random values from the Gaussian distribution. The space in which all z-vectors are located is called Z-space and is denoted by Z. The output of the network is the vector w which belongs to the latent space W. Latent space is an

abstract multidimensional space that contains characteristic values that cannot be directly interpreted. The second part of the generator is a network that generates an image with the help of a constant vector, random noise and vector w transformed by affine transformations.

The vector w transformed by affine transformations serves to adjust the style of synthetically generated images using adaptive instance normalization, AdaIN. Image style represents a vector in latent space that is separated from the semantic content vector of the image [20]. An example of transferring a style from one image to another is given in the figure 3.15.

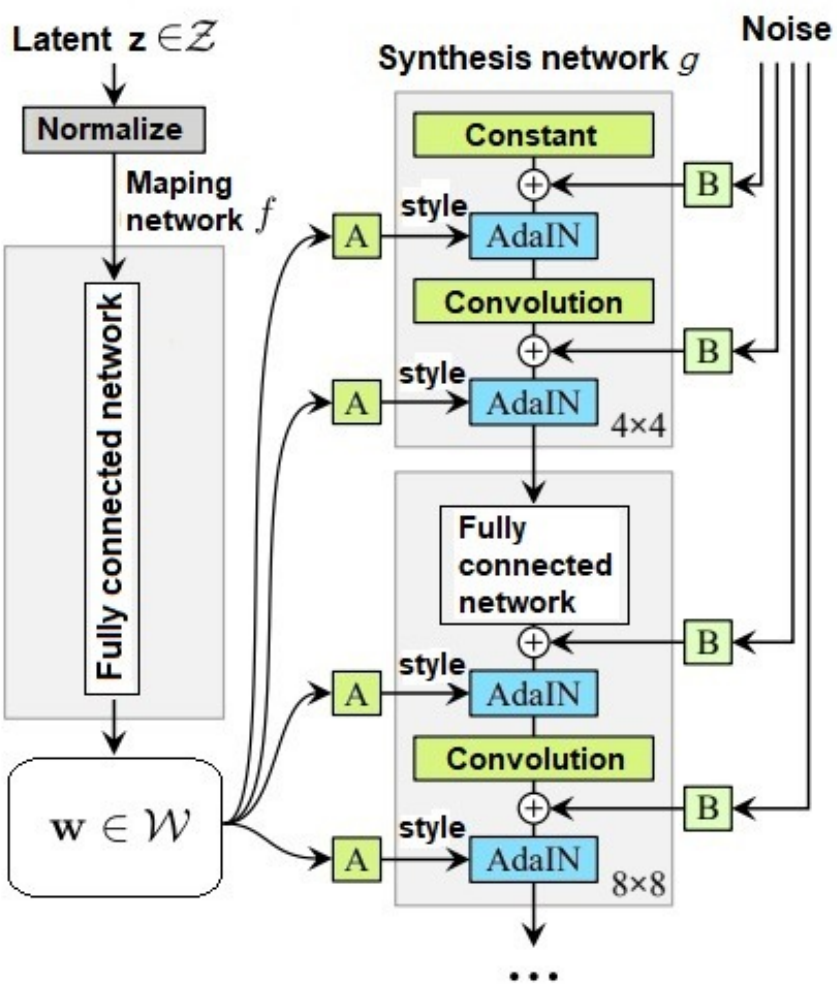

Figure 3.14: A style-based generator model, "A" represents affine transforms while "B" represents a scaling factor for noise [23].

The *StyleGAN* model generates photorealistic images. During development, the *StyleGAN* model went through a phase of optimization and minor changes to the

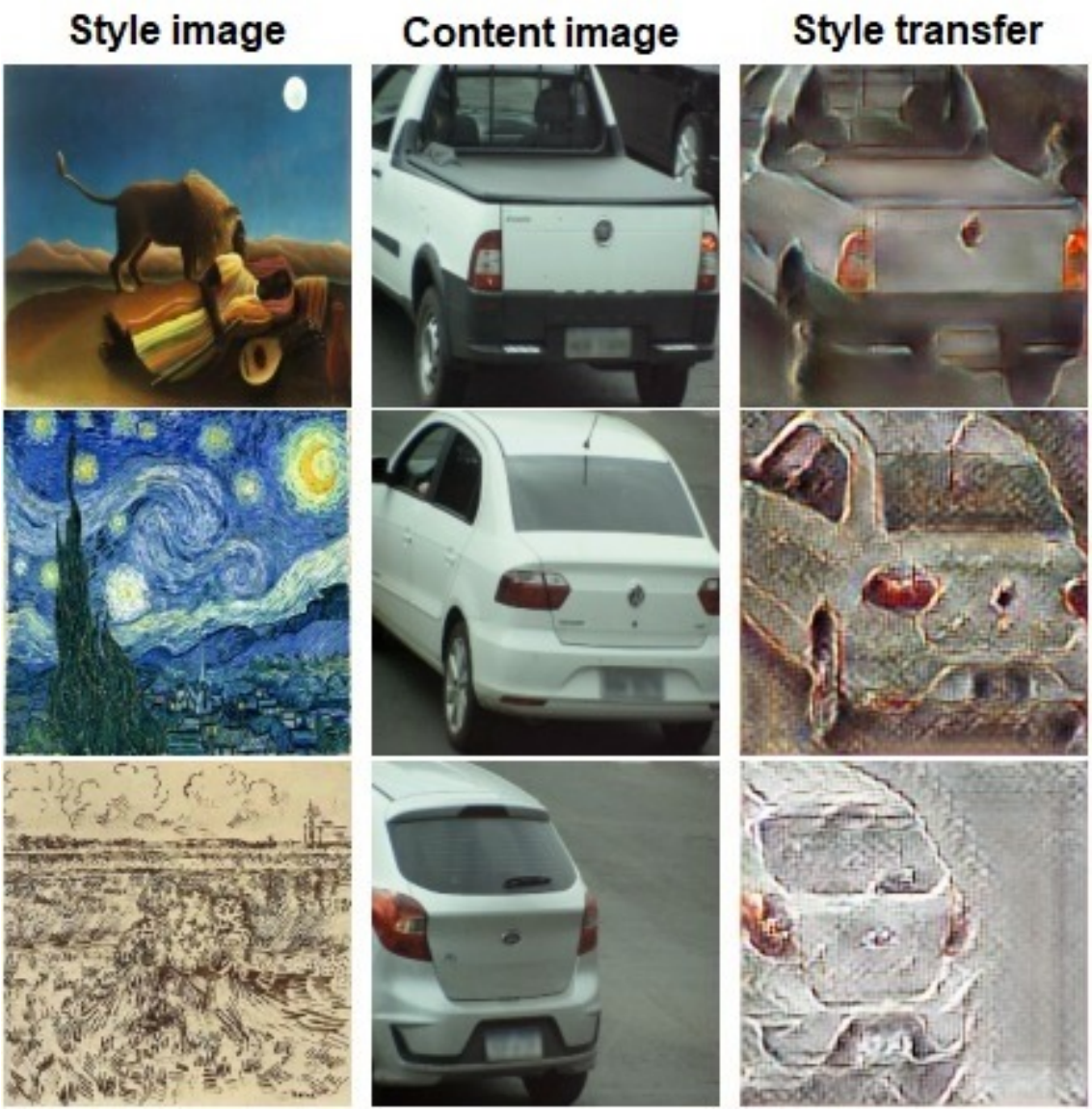

Figure 3.15: Transfer of styles to desired images and results of AdaIN operation [20].

synthesis network. The new version of *StyleGAN*, *StyleGAN2* brings up to 60% faster model training compared to the first version [24]. Due to such speedups, the processing power required for training is reduced. The paper uses this improved version, more precisely the *StyleGAN2* architecture.

Unless a large amount if data is available, working with generative adversarial networks can result in low-quality models. In order to alleviate such a problem, the method of differentiable augmentation was introduced for the *StyleGAN2* model. The method uses different types of augmentation on real and fake data. The method

enables more stable training and leads to better convergence compared to training the model without the differentiable augmentation method over the same dataset. The *StyleGAN2* model with the differentiable augmentation method is able to produce realistic images, even over a dataset containing only one hundred images [37].

3.4 Transfer Learning Technique

In supervised learning, models are trained to learn the relationships that hold between given inputs and outputs. The models can then make output predictions on the new data. Models developed in this way will perform better when solving a problem that is in the same domain as the training data. The result will deteriorate if the problem domain changes. In the worst case, it may be necessary to train a new model even when the problem domains are similar [38].

The transfer learning technique is often used when training a model to solve a new problem requires a large amount of resources. Using the transfer learning technique brings a number of advantages, the main of which are resource saving and error reduction during model training [38]. The transfer learning technique takes relevant parts of an existing learned model and applies them to solve similar problems.

The transfer learning technique is one of the techniques used to overcome the problem of not having enough data. Transfer learning transfers knowledge from an existing model to a new model that should solve a similar task. A key part of transfer learning is generalization. This means that only knowledge that can be used in other domains is transferred. Instead of models being tightly bound to a training dataset, the models used in transfer learning are more general [38].

Chapter 4

Model evaluation techniques

To evaluate the model in object detection, the mean average precision (mAP) measure is used, which takes into account precision (precision) and recall. Also, a confusion matrix is used, where the calculation of correctly and incorrectly classified objects depends on the given threshold of the intersection over union metric IoU and the confidence score of classified objects. The confidence score represents the object's probability value that the model assigned to it during classification.

4.1 Intersection over Union

For object detection tasks, precision is calculated using the Intersection over Union metric. IoU is an evaluation metric used to measure the accuracy of an object detector on a given dataset. The picture 4.1 shows an example of a frame, where the intended frame is drawn in red, while the real frame is drawn in yellow. The metric is calculated by dividing the intersection of two frames by their union (figure 4.2).

4.2 Confusion matrix for object detection

The confusion matrix is a tabular representation of the numbers of correctly and incorrectly classified objects, on the basis of which classification models can be evaluated. For multi-class classification of object detection, the confusion matrix is expanded by one column and one row. The new column serves to record the number of objects that are classified in the image, but are not on it. The row is used to record the number of objects that are in the image, but are not classified. Objects

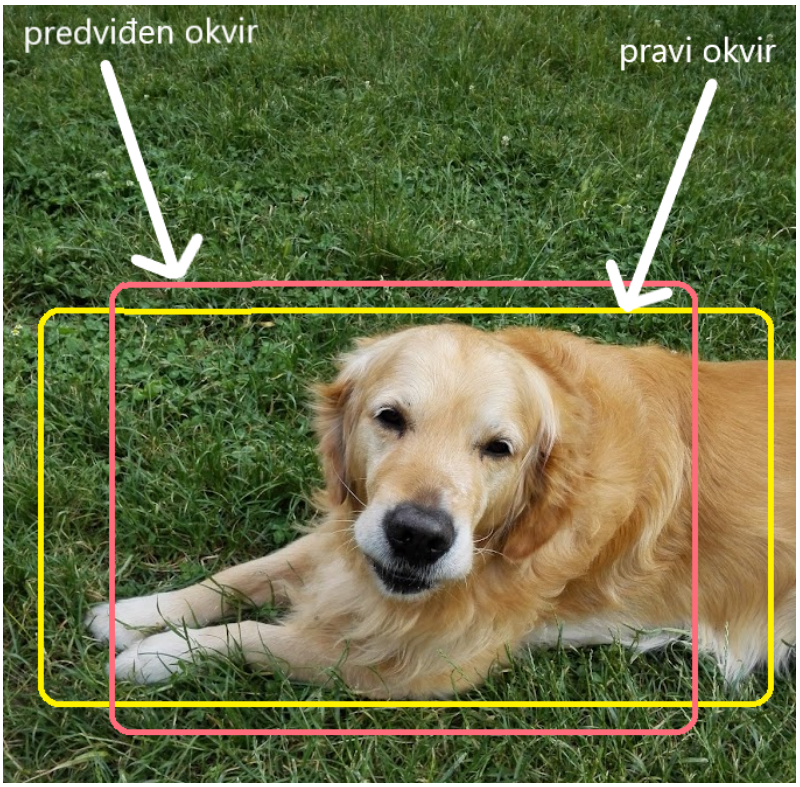

Figure 4.1: Predicted frame (red) and real frame (yellow) over given object

Figure 4.2: IoU metric calculation demonstration

are classified as correct or incorrect based on both the IoU threshold (usually 0.5 is used in practice) and the condfidence score threshold.

The confusion matrix is also calculated for each class individually and is similar to binary classification. When calculating the confusion matrix, one specific class is observed and that class is considered positive, while the set containing all objects of the other classes is considered a negative class. The confidence score does not participate in the computation of the confusion matrix for each class.

True positives (TP) is the number of objects that belong to the positive class and were assigned a positive class by the model. Also, the IoU value of each object is greater than the set threshold.

True negative (TN) is the number of objects that do not belong to the positive class and the model did not assign a positive class to them. IoU doesn't count, because if the objects don't exist, their bounding boxes won't exist either.

False positives (FP) is the number of objects that belong to the negative class but were assigned a positive class by the model. Also, they can be objects that belong to the positive class and the model assigned them a positive class, but the IoU value of each object does not exceed the given threshold.

False negatives (FN) is the number of objects that belong to the positive class and the model assigned them a negative class. The IoU is not compared to the threshold because the classification is wrong.

4.3 Precision, recall and F_1-score

Precision is the percentage of truly positive objects among objects that are classified as positive and is defined by the formula:

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

Higher precision also means that there are fewer false positive objects.

Recall represents the percentage of positive objects that are correctly classified and is defined by the formula:

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

A high recall means that there are few false negative objects.

Precision and recall can be summed up in a measure, which is called the F_1-score and is defined by the formula:

$$F_1 = 2\frac{Precision \cdot Recall}{Precision + Recall} \tag{4.3}$$

The macro-averaged F_1-score is calculated as an arithmetic mean:

$$F_{1avg} = \frac{1}{n}\sum_{k=0}^{k=n-1} F_{1k} \tag{4.4}$$

where F_{1k} is the value of the F_1-measure for the kth class.

The average weighted F_1-measure is calculated by the following formula:

$$F_{1wavg} = \frac{1}{n} \sum_{k=0}^{k=n-1} (F_{1k} S_k) \tag{4.5}$$

where F_{1k} and S_k are the value of the F_1-measure and the total number of instances in the kth class.

4.4 Mean Average Precision

Average precision is a method of calculating the area under the precision and recall curve for one class. Precision and recall values are calculated for given values of IoU thresholds from the interval [0.5 to 1.0) with a common step of 0.05. The range of calculated recall and precision values are extended by **0** and **1**, respectively.

Average precision is calculated using the following formula:

$$AP = \sum_{k=0}^{k=n-1} (R_k - R_{k+1}) P_k \tag{4.6}$$

where R_k and P_k are the recall and precision for the kth IoU threshold. The mean average precision is calculated as the mean value of all the average precisions given by the formula:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{4.7}$$

where N represents the number of classes and AP_i the average precision for a class i.

4.5 Fréchet inception distance

Fréchet inception distance, FID, is a metric used to evaluate the quality of images generated by generative adversarial networks [17]. The Fréchet inception distance compares the distribution of the generated images to the distribution of the actual images that were used to train the generator. The Fréchet inception distance for two identical images gives a value of zero, while for different ones it gives a number greater than zero.

The Fréchet inception distance is calculated for two joint normal distributions $\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu', \Sigma')$by the following formula [8]:

$$d_F(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu', \Sigma'))^2 = \|\mu - \mu'\|_2^2 + tr\left(\Sigma + \Sigma' - 2\left(\Sigma^{\frac{1}{2}} \cdot \Sigma' \cdot \Sigma^{\frac{1}{2}}\right)^{\frac{1}{2}}\right) \tag{4.8}$$

In practical use, image distributions are obtained by computing mean values (μ, μ') and covariance matrices (Σ,Σ') using the vectors obtained by the model *InceptionV3*[1].

4.6 Loss function for generative adversarial networks

The loss function for generative adversarial networks can be formulated as a zero-sum game [14, 18]. The generator tries to minimize the function while the discriminator tries to maximize. For a given generator G and discriminator D, the zero-sum game problem is defined as [14, 26]:

$$\min_G \max_D \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[1 - \log D(G(z))] \tag{4.9}$$

In the given formula:

- $D(x)$ represents the discriminator's estimate that the real data x is real,
- $\mathbb{E}_x$ is the expectation from the distribution of real data,
- $G(z)$ is the output of the generator for the value z from the latent space,
- $D(G(z))$ represents the discriminator's estimate that the false data is real,
- $\mathbb{E}_z$ is the expectation from the data distribution in the latent space.

The loss function for the generator is defined by:

$$\min \log\left(1 - D(G(z))\right) \tag{4.10}$$

and for the discriminator with:

$$\max \log D(x) + \log\left(1 - D(G(z))\right) \tag{4.11}$$

[1]InceptionV3 is a convolutional neural network used for image analysis and object detection.

Chapter 5

Implementation and results

As part of the implementation, the available dataset is expanded with augmentation methods and synthetic data generated by the *StlyeGAN2* model. In order to assess whether the applied techniques enable improvement, the model is trained on the basic data set and compared with the models trained on the extended dataset. The code developed as part of the thesis work is available at *github* for object detection [16]. The data used and generated in this work are available at at this address [15]. The *Google Colab* platform was used for model training. The platform provided an NVIDIA Tesla P100-PCIE graphics card with 16GB of RAM.

5.1 Available data

The labeled dataset named Brazilian vehicle set, BVS for short, was used in this work and was provided by the institute BRAIN (Brazilian Atrifical Intelligence Nucleus), faculty FASCENS (pt. *Faculdade de Engenharia de Sorocaba*, São Paulo, Brazil. The dataset BVS contains four classes: *car*, *motorbike*, *truck* and *bus*. Examples from these classes are shown in the figure 5.1. The total amount of data is 1872 images, most of which belong to the *car* and *motorbike* classes. In the dataset BVS, the resolution of all images is 800×600px.

The images are divided into two sets, a training set and a validation set. 10% images from each class were randomly taken for the validation set. The validation set did not participate in any augmentation technique.

The total number of images of each class, the number of images belonging to the training set and the number of images belonging to the validation set are shown

Table 5.1: The total number of images for each class, and the number after splitting into a training set and a validation set

class	total	training set	validation set
car	889	800	89
motorbike	966	869	97
truck	10	9	1
bus	7	6	1

(a) *car* (b) *motorbike*

(c) *truck* (d) *bus*

Figure 5.1: Examples of images from available classes

in the table 5.1. From the table 5.1 it can be seen that the imbalance between the classes is extremely high, even 1:15 between the classes *bus* and *car*.

5.2 Creating Synthetic Data

The training set was augmented in two ways: images were generated using standard augmentation methods and the *StyleGAN2* model. Table 5.2 shows the exact number of images that were generated in order to obtain balanced classes, more precisely that each class contains an equal number of images in the training set. The number of images chosen to expand the training set was chosen based on the class that contained the largest amount of data, specifically the *motorbike* class. The *motorbike* class is expanded with 869 images, more precisely with as many images as are present in the training set. After expanding the training set, each class contains 1738 images.

Table 5.2: Total number of generated images

class	car	motorbike	truck	bus
number of images generated	938	869	1732	1729

Data generation by standard augmentations

Due to the need to expand the dataset BVS, new data are generated by augmentation techniques (chapter 3.2). The extended set is called BVSstandard later in the paper. The set of augmentations that are applied are:

- flipping — is performed only in the horizontal direction and care is taken to change the coordinates of the border frames,
- rotation — was done with a random angle between one and five degrees and the empty space is filled with border pixel colors,
- translation — moves the image in a random direction by ten pixels and the empty space is filled with the colors of the border pixels,
- Gaussian Noise — adds noise to the image with a constant standard deviation of 0.1 and a mean of 1.0,
- Gaussian Blur — is applied to the image with a constant standard deviation of 7. The standard deviation value is chosen because of the resolution of the images in the BVS set. Using a value of 7 for the standard deviation blurs the images while still making it possible to recognize the object in the image.

- color augmentation — The brightness parameter is randomly selected from a uniform distribution from the interval 0.5 to 1.5. Contrast, saturation and hue parameters are randomly selected from a uniform distribution from the interval 0.0 to 10.0. Color augmentation intervals were chosen after testing intervals of different ranges. It was concluded that the best results are obtained using the specified intervals.

Four randomly chosen augmentations are applied from the described set of augmentations. One affine transformation is chosen at random from the set consisting of flipping, rotation and translation. Also, one augmentation is chosen between Gaussian blur and Gaussian noise. From the set of color augmentations that make up brightness, contrast, saturation and hue, two augmentations are randomly selected. Examples of generated images are shown in figure 5.2.

Figure 5.2: Examples of images generated by standard augmentations

Data generation with the *StyleGAN2* model

The dataset BVS is extended with synthetic data generated by the *StyleGAN2* model. The name BVSstyle is used for this extended set in the paper.

However, the problem of small amount of data also occurs for the generation of new images by the *StyleGAN2* model. Using the differentiable augmentation method outlined in section 3.3 overcomes the problem of small amount of data and the model successfully generates images. Implemented method dof differentiable augmentations was tested on the CIFAR-10 dataset [36].

The *StyleGAN2* model architecture is implemented using the *Tensorflow* 2.0 library[1]. The generated images are 512×512 in size. The models were trained on the NVIDIA Tesla P100-PCIE graphics card for twelve to forty-eight hours, depending on the class itself. Every six hours of training or, more precisely, every twenty-five thousand epochs, a checkpoint was made. After each checkpoint, images were generated to calculate the Fréchet inception distance.

The Fréchet inception distance metric and loss function for generative adversarial networks were used to evaluate the *StyleGAN2* model. After each validation point, Fréchet inception distance was used to assess the similarity between the two sets of images: the images generated by the model were compared to the real images from the validation set BVS. The calculation of the Fréchet inception distance took place on an NVIDIA 780Ti graphics card and would take a total of ten minutes. When an object could be recognized in the generated image and when the loss function of the discriminator and generator did not decrease significantly, further training of the model was stopped. The exact number of epochs of each class is given later in the text.

Classes *Bus* and *Truck*

Training the *StyleGAN2* model on the *truck* and *bus* classes was a big challenge due to the small amount of available data. The Fréchet inception distance results are given in tables 5.3, 5.4.

The training of the model lasted twenty-four hours. Two hundred thousand images were generated for model training using the differentiable augmentation

[1]The *Tensorflow* 2.0 library is open source and provides a comprehensive ecosystem of tools. It was published in 2015 by the *Google Brain* team. The main purpose is to create machine learning models. The library is intended for use in the programming language *Python*.

Table 5.3: Class *truck*

epoch number	FID
25 thousand	432.8
50 thousand	188.5

Table 5.4: Class *bus*

epoch number	FID
25 thousand	369.2
50 thousand	220.5

method. The original training image set for the *bus* class contained six images, and the training set for the *truck* class contained nine images.

From the loss function (figure 5.3) and the Fréchet inception distance result (table 5.3) for the *truck* class, it can be concluded that further training for the next twenty-five thousand epochs does not they get significantly better results.

Examples of generated images of the *truck* class are shown in the figure 5.5. For the *bus* class, from the loss function (image 5.4) and the result of the Fréchet inception distance (table 5.4), it can be concluded that further training for the next twenty-five thousand epochs does not they get significantly better results. Examples of generated images of the *bus* class are shown in the figure 5.6.

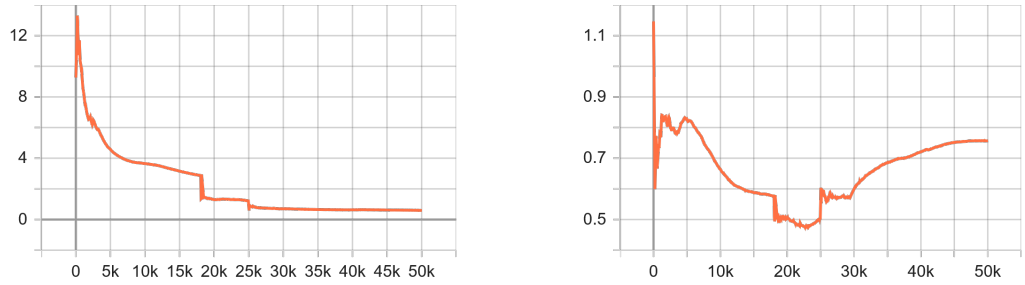

Figure 5.3: L functions for the *truck* class, generator and discriminator. In both cases, the x axis represents the epochs, while the y axis represents the values of the corresponding loss function.

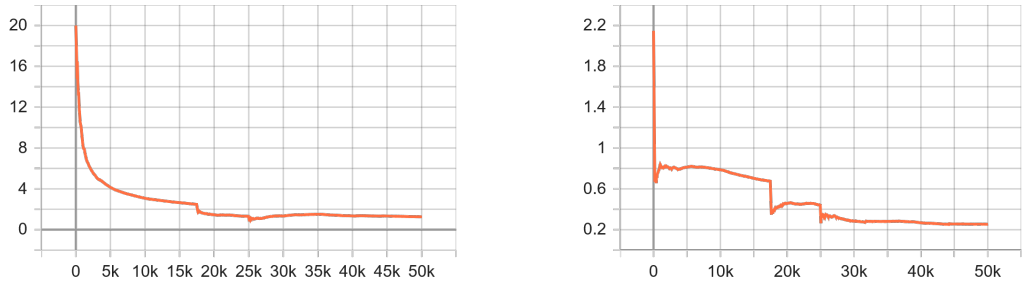

Figure 5.4: Loss functions for class *bus*, generator and discriminator. In both cases, the x axis represents the epochs, while the y axis represents the values of the corresponding loss function.

Figure 5.5: Examples of generated images of the class *truck*

Figure 5.6: Examples of generated images of the class *bus*

Classes *car* and *motorbike*

The training of the model over the *car* class lasted about thirty-six hours and three hundred thousand images were generated using the differentiable augmentation method. The training for the *motorbike* class lasted forty-eight hours and four hundred thousand images were generated using the differentiable augmentation method.

For the *car* class, the loss function is shown in figure 5.7. The result of the Fréchet inception distance is shown in table 5.5. Further training of the *car* class does not result in significant improvements to the model. Examples of generated images of the *car* class are shown in the figure 5.9.

The *motorbike* class has slightly worse results with the loss function shown in figure 5.8. The generator's loss function is increasing, indicating that the discriminator is getting much better at distinguishing real from fake images. The images generated by the generator are not perfect, but it is possible to recognize the main characteristics of objects that are important for training object detection models. The result of the Fréchet inception distance (table 5.6) indicates that during training there is an improvement in the generated images. Examples of generated images of the *motorbike* class are shown in figure 5.10.

Table 5.5: Class *car*

epoch number	FID
25 thousand	107.6
75 thousand	87.5

Table 5.6: Class *motorbike*

epoch number	FID
25 thousand	386.8
100 thousand	151.1

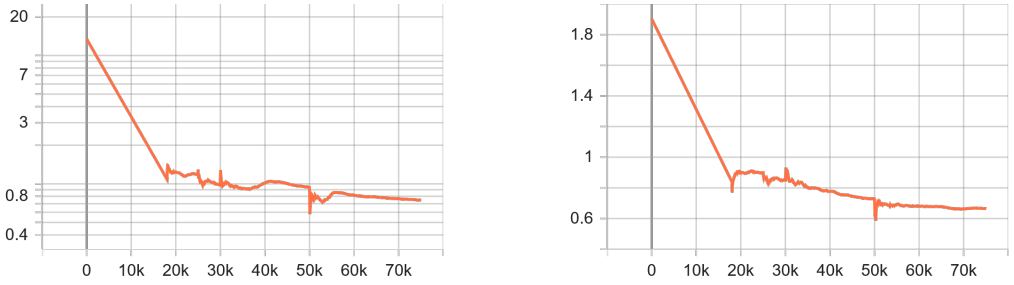

Figure 5.7: Loss functions for class *car*, generator and discriminator. In both cases, the x axis represents the epochs, while the y axis represents the values of the corresponding loss function.

The generated images are not perfect, but it is possible to recognize the main features of the objects. Some generated images tend to contain artifacts and blurs.

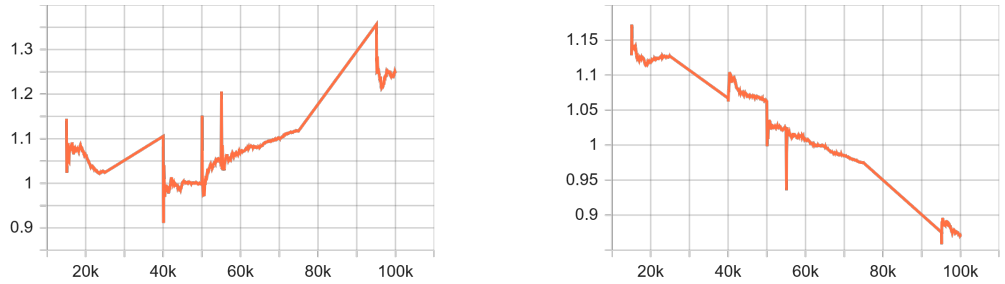

Figure 5.8: Loss functions for the *motorbike* class, generator and discriminator. In both cases, the x axis represents the epochs, while the y axis represents the values of the corresponding loss function.

Figure 5.9: Examples of generated class images *car*

5.3 Model for object detection and prediction VGG16

The basic VGG16 model is limited to image classification without bounding box prediction and in this work, the basic model is extended with appropriate networks to enable bounding box prediction. The basic model is extended as follows:

- Added a fork to two new fully connected networks of depth four on the last layer.
- At the output of the first network, an activation function *softmax* was added, which is interpreted as assigning the probability that the object belongs to one of the four classes.

Figure 5.10: Examples of generated images of the *motoribke* class

- At the output of the second network, a *sigmoid* activation function was added, which is interpreted as assigning relative bounding boxes to the intended objects.

The extended base model was trained on the BVS dataset on NVIDIA Tesla P100-PCIE graphics card for about ten minutes.

Due to the class imbalance problem that occurs in the BVS dataset, the result of the model is highly biased towards the *car* class, as can be seen from the 5.7 table. The model outputs the bounding boxes incorrectly, as shown in figure 5.12.

The results of the model evaluation are given in the table 5.8. The results show that the model behaves poorly over the entire dataset. Some of the reasons that influence the model to give bad results are the high imbalance between classes, the small amount of available data, and the model is too simple and cannot handle all the possible features found in the images. The precision of the classes in relation to the IoU threshold is shown in the figure 5.11.

As the basic VGG16 model is extended, there are no suitable weights of another model that can be used for the transfer learning method. Using the extended dataset BVSstandard or BVSstyle, the amount of data available would still be insufficient. Tens of thousands of images are needed for model training, which are not even available in the extended BVS sets.

Table 5.7: Confusion matrix for VGG16 with IoU threshold 0.5 and confidence threshold for recognized class 0.5

	car	motorbike	bus	truck	without detection
car	85	26	0	1	0
motorbike	0	0	0	0	0
bus	0	0	0	0	0
truck	0	0	0	0	0
without detectio	4	71	1	0	0

Table 5.8: VGG16 model results

class	average precision (eq. 4.6)	mean F_1-measure (eq. 4.4)
car	0.95	0.57
motorbike	0.0	0.0
bus	0.0	0.0
truck	0.0	0.0
Mean average precision (eq. 4.7)	0.24	
Mean weighted F_1-measure (eq. 4.5)	0.27	

5.4 YOLO object detection and prediction model

As the VGG16 model cannot provide satisfactory results, the YOLO object detection and prediction model is also considered in the work. Two configurations of the YOLO model are used. The first configuration of the YOLO model serves to compare the results, and is able to predict up to eighty classes. Another configuration of the YOLO model is able to predict up to four classes.

Prediction of the YOLO model trained on the COCO dataset

The YOLO model is used, which is trained on the common objects in context dataset (COCO), which contains eighty classes and 328 thousand images. Further in the paper, the YOLO model trained on the COCO dataset is named as YOLO_{coco}. Model evaluation is performed on the validation data set BVS. The confusion

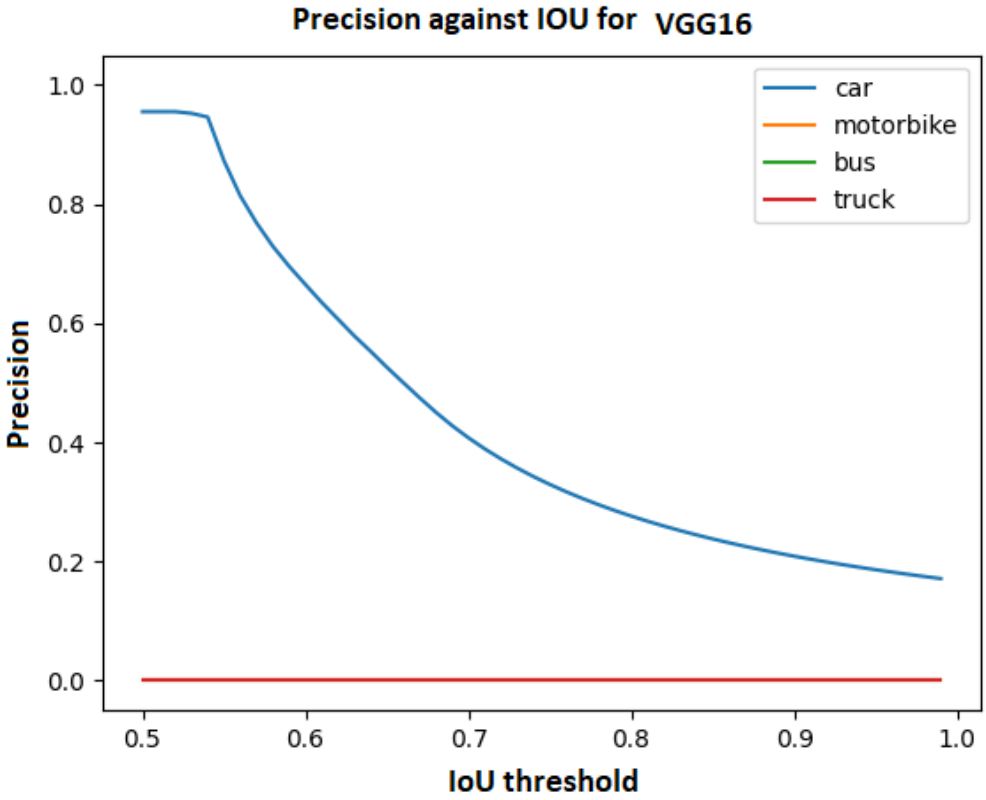

Figure 5.11: Precision in relation to the IoU threshold for the VGG16 model, for classes *motorbike*, *car* and *bus* the precision in all IoU values is 0

matrix was calculated for an IoU threshold value of 0.5 and for a confidence score greater than 0.5. During the calculation of the average precision and mean F_1-measure, the condfidence score of the classes was not taken into account.

From the results of the confusion matrix (table 5.9) and the model evaluation method (table 5.10), the following can be concluded:

- The class *car* is predicted by the YOLO_{coco} model with a high precision. Eighty-one instances out of a total of eighty-nine are predicted. From the confusion matrix, it can be seen that the YOLO_{coco} model for three instances had a condfidence score of less than 0.5. It also misclassifies five instances as instances of the *truck* class. The average precision for the *car* class is 0.88, while the mean F_1-measure is 0.91.

- The *motorbike* class is predicted worse than the *car* class. Model YOLO_{coco} predicts thirty-seven of the total ninety-seven instances. The majority of in-

(a) Actual Frames (b) Expected frames

Figure 5.12: Example results of the VGG16 model over the validation dataset

Table 5.9: Confusion matrix for YOLO_{coco} with IoU threshold 0.5 and confidence threshold for recognized class 0.5

	car	motorbike	bus	truck	without detection
car	81	0	0	0	0
motorbike	0	37	0	0	0
bus	0	0	0	0	0
truck	5	0	0	1	0
without detection	3	52	1	0	0

stances, forty-nine to be exact, have a value of less than 0.5 for the probability rating. The likelihood class scores indicate that the model is not confident in predicting instances from the *motorbike* class. The average precision is 0.77 and the mean F_1-measure is 0.48.

- The *bus* class contains one validation image and it is not predicted by the YOLO_{coco} model. The average precision and mean F_1-measure are 0.
- The class *truck* is successfully predicted by the YOLO_{coco} model. The average precision is 1.0 and the mean F_1-measure is 0.99.

The precision of the classes in relation to the IoU threshold is shown in the figure 5.13. The YOLO_{coco} model has a mean average precision (mAP) of 0.66 and a mean weighted F_1-measure of 0.69. When predicting the YOLO_{coco} model, we can conclude that it recognizes the *car* and *truck* class objects well. The problem observed when predicting those classes is that there are objects that have similar characteristics. Such a problem can be seen in the image 5.15. The YOLO_{coco} model successfully classified 40% of the instances of the *motorbike* class from the validation set BVS. An instance of the class *bus* is not correctly classified by the YOLO_{coco} model.

Model training using the transfer learning technique

To improve the results, subsequent models were trained using the weights of the models trained on the COCO dataset. As there is a difference in the number of classes between the BVS dataset and the COCO dataset, some parts of the weights are discarded.

Models are trained on images with a resolution of up to 512×512. The images are randomly resized from the interval 224×224 to 512×512, with a step of 32×32.

Table 5.10: Results of model evaluation YOLO_{coco}

class	average precision (eq. 4.6)	mean F_1-measure (eq. 4.4)
car	0.88	0.91
motorbike	0.77	0.48
bus	0.0	0.0
truck	1.0	0.99
Mean average precision (eq. 4.7)	0.66	
Mean weighted F_1-measure (eq. 4.5)	0.69	

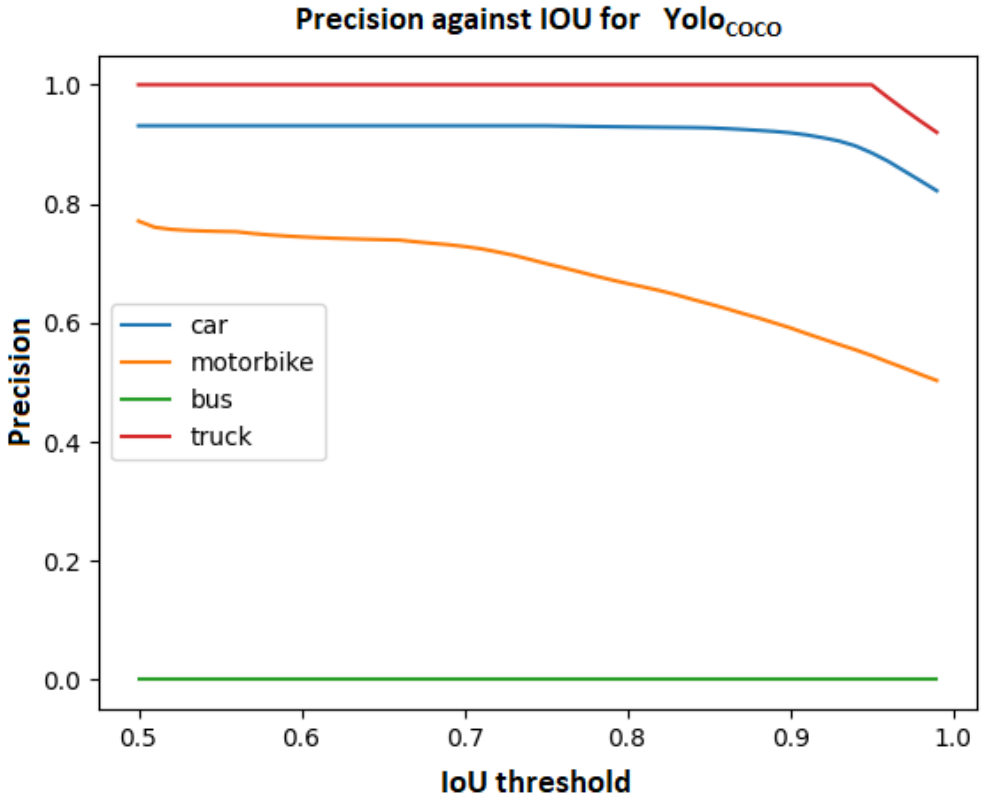

Figure 5.13: Precision in relation to the IoU threshold [0.5,1) of the model YOLO_{coco}

Model training was limited to one hundred and ten epochs. In the first ten epochs, model training was performed in only the last three layers, while the other layers were frozen. After ten epochs, the remaining layers were unfrozen and normal model training continued. In the first ten epochs, the training rate of the model

Figure 5.14: Images from the validation set and predictions over them by the YOLO$_{coco}$ model, class *car* (top), class *motorbike* (middle), class *truck* (bottom left) and the *bus* class (bottom right).

Figure 5.15: Similar features of two different classes, class *car* (left) and class *truck* (right)

optimizer was decreased every three epochs by a constant value of 0.001, in case the loss function was not decreasing. After ten epochs the training rate of the model optimizer decreased by 0.00001. This allowed the improvement of the model. Also, an early stop flag is set if the loss function has not decreased in ten consecutive epochs.

Model training using the transfer learning technique on the BVS dataset

The YOLO model is used which is trained on the BVS dataset containing four classes and 1872 images. Further in the paper, the YOLO model trained on the data set BVS is denoted by YOLO_{BVS}. The model loss function is shown in Fig. 5.16. It can be seen from the loss function that the model training stopped early after twenty-seven epochs. The model was trained on an NVIDIA Tesla P100-PCIE graphics card for about two hours.

From the results of the confusion matrix (table 5.11) and the model evaluation method (table 5.12), the following can be concluded:

- class *car* is successfully classified by the YOLO_{BVS} model in 66% of cases. Fifty-nine out of a total of eighty-nine instances were predicted. From the confusion matrix, it can be seen that the YOLO_{BVS} model for thirty instances has a condfidence score of less than 0.5. The average precision for the *car* class is 0.06 and the mean F_1-measure is 0.04, indicating very poor results.

- ten instances of the *motorbike* class are incorrectly predicted. The YOLO_{BVS}

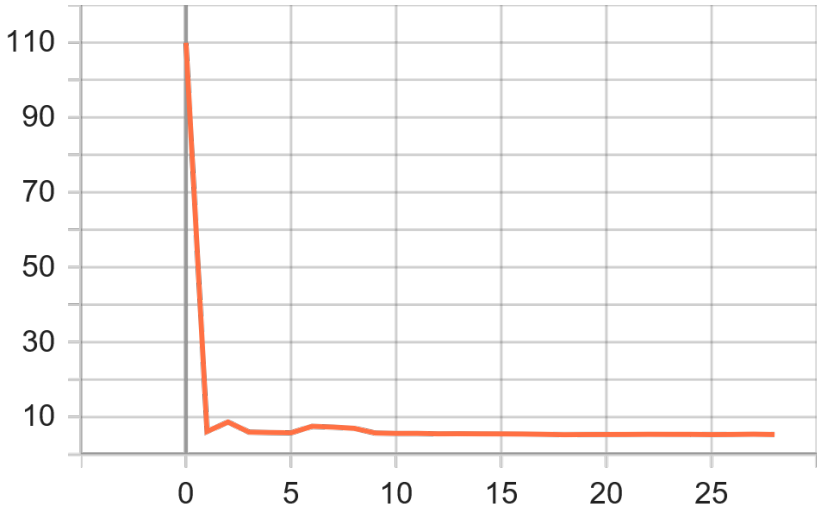

Figure 5.16: The loss function of the YOLO_{BVS} model

Table 5.11: Confusion matrix for YOLO_{BVS} with IoU threshold 0.5 and confidence threshold for recognized class 0.5

	car	motorbike	bus	truck	without detection
car	59	10	0	0	0
motorbike	0	0	0	0	0
bus	0	0	0	0	0
truck	0	0	0	0	0
without detections	30	79	1	1	0

model predicted them as instances of the *car* class. The rest of the *motorbike* class is not predicted. The average precision and mean F_1-measure are 0.

- class *bus* and class *truck* are not predicted in any instance. The average precision and the mean F_1-measure are also 0.

The precision of the classes in relation to the IoU threshold is shown in the figure 5.17. The YOLO_{bvs} model has a mean average precision (mAP) of 0.01 and a mean weighted F_1-measure of 0.02. One can complain without further analysis that the model behaves badly and that the predictions are bad. An example of the detection results is shown in the image 5.18. It can be seen from the images that the model classifies only the *car* class and that very poorly with a large number of bounding boxes.

Table 5.12: Results of model evaluation YOLO_{bvs}

class	average precision (eq. 4.6)	mean F_1-measure (eq. 4.4)
emperor	0.06	0.04
motorbike	0.00	0.00
bus	0.00	0.00
truck	0.00	0.00
Mean average precision (eq. 4.7)	0.01	
Mean weighted F_1-measure (eq. 4.5)	0.02	

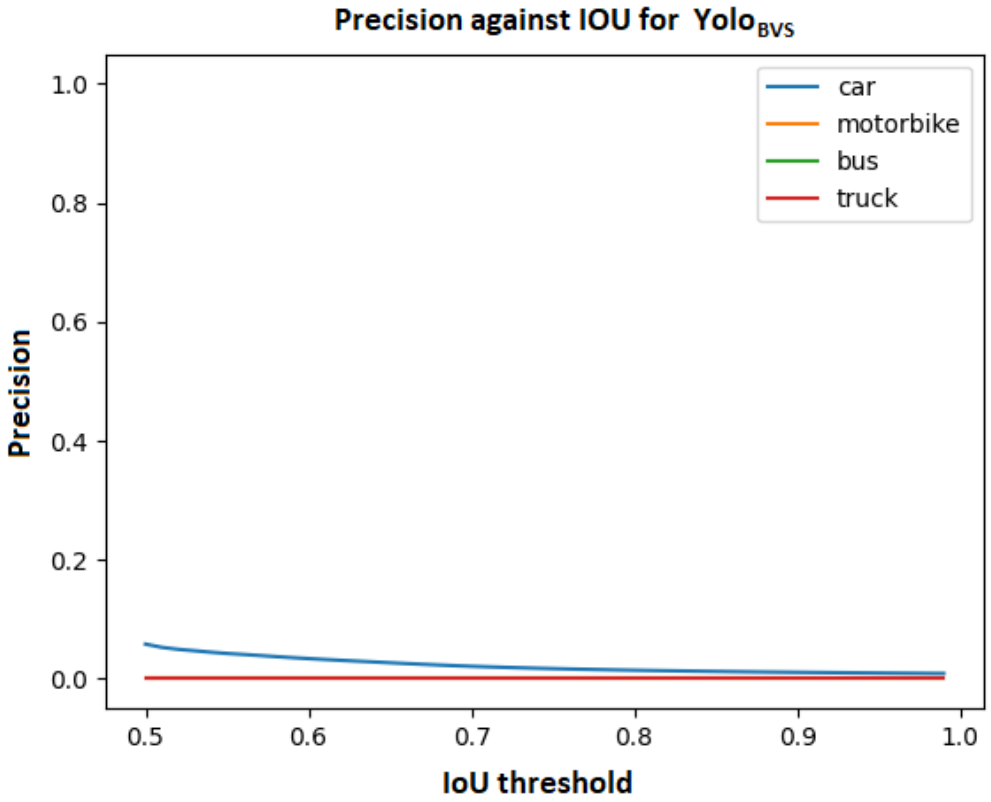

Figure 5.17: Precision with respect to the IoU threshold [0.5,1) of the YOLO_{BVS} model, for classes *motorbike*, *bus* and *truck* precision in all IoU values is 0

Model training using the transfer learning technique on the data set BVSstandard

The YOLO model is used, which is trained on the BVSstandard dataset containing four classes and 6952 images. Further in the paper, the YOLO model trained on the

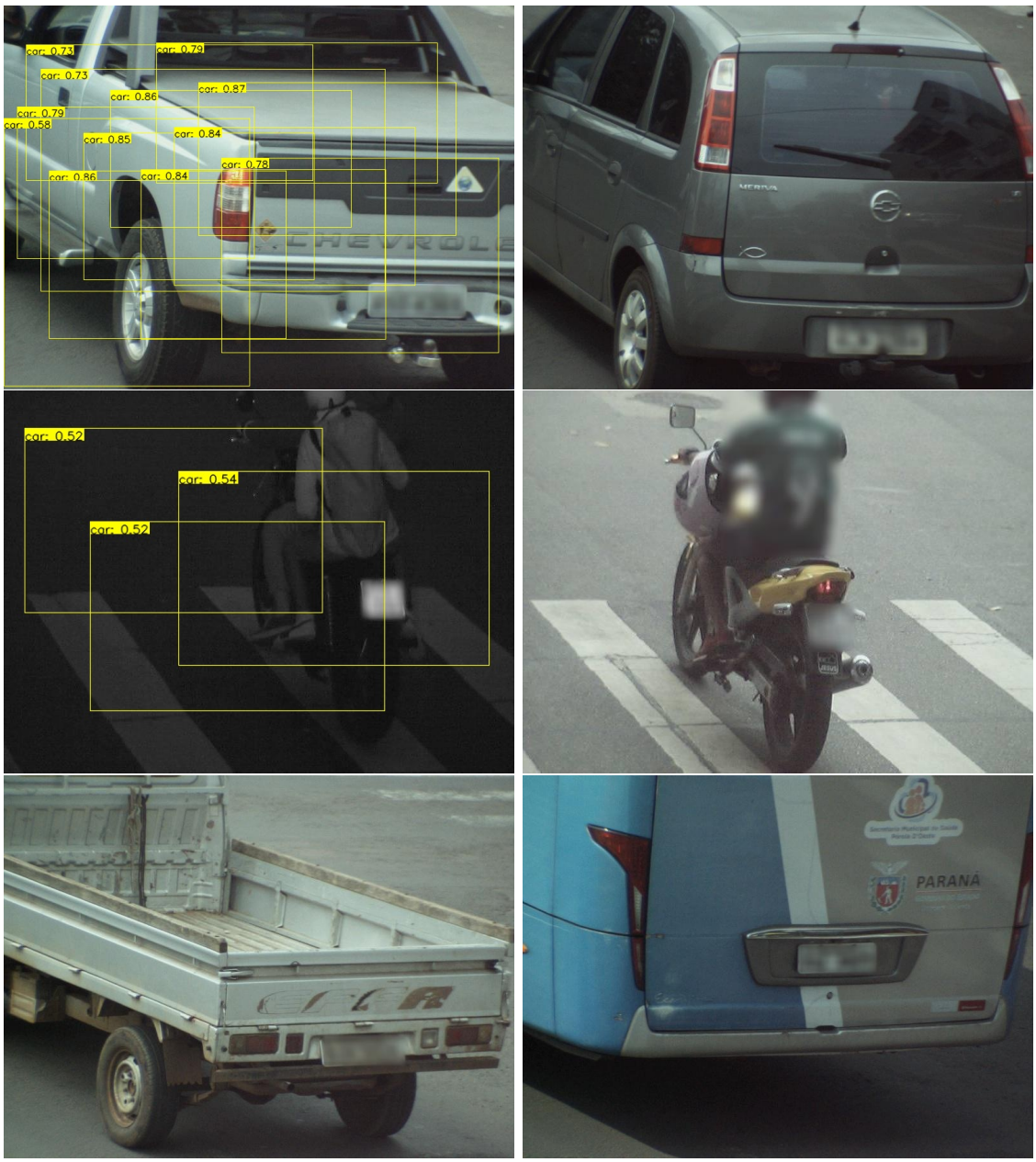

Figure 5.18: Images from the validation set and predictions over them by the YOLO_{BVS} model, class *car* (top), class *motorbike* (middle), class *truck* (bottom left) and the *bus* class (bottom right). Images without bounding boxes are not successfully classified by the YOLO_{BVS} model.

data set BVSstandard is denoted by YOLO$_{\text{BVS}standard}$. The model loss function is shown in figure 5.19. It can be seen from the loss function that the model training stopped early after forty epochs. The model was trained on an NVIDIA Tesla P100-PCIE graphics card for eleven hours.

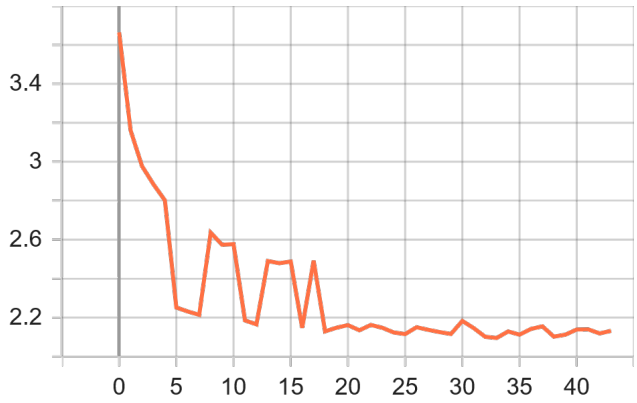

Figure 5.19: Loss function of the YOLO$_{\text{BVS}standard}$ model

From the results of the confusion matrix (table 5.13) and the model evaluation method (table 5.14), the following can be concluded:

- class *car* is predicted by the YOLO$_{\text{BVS}standard}$ model with a high precision. Eighty-two out of a total of eighty-nine instances were predicted. From the confusion matrix, it can be seen that the YOLO$_{\text{BVS}standard}$ model had a condfidence score of less than 0.5 for seven instances. The average precision for the *car* class is 1.0[2], while the mean F_1- the measure is 0.92. These results indicate a very good prediction of the YOLO$_{\text{BVS}standard}$ model over the *car* class.

- class *motorbike* is predicted better than class *car*. The YOLO$_{\text{BVS}standard}$ model predicts ninety-five of the total ninety-seven instances. Two likelihood rating instances have a value less than 0.5. The average precision is 0.99 and the mean F_1-measure is 0.97.

- class *truck* is incorrectly predicted as the class *car*.

- class *bus* is not predicted by the YOLO$_{\text{BVS}standard}$ model.

[2]The condfidence score was not used to calculate the average precision and mean F_1-measure, but only whether the object was successfully classified or not.

Table 5.13: Confusion matrix for YOLO$_{\text{BVS}standard}$ with IoU threshold 0.5 and confidence threshold for recognized class 0.5

	car	motorbike	bus	truck	without detection
car	82	0	0	1	0
motorbike	0	95	0	0	0
bus	0	0	0	0	0
truck	0	0	0	0	0
without detection	7	2	1	0	0

Table 5.14: Model results YOLO$_{\text{BVS}standard}$

class	average precision (eq. 4.6)	mean F_1-measure (eq. 4.4)
car	1.0	0.92
motorbike	0.99	0.97
bus	0.00	0.00
truck	0.00	0.00
Mean average precision (eq. 4.7)	0.50	
Mean weighted F_1-measure (eq. 4.5)	0.93	

The precision of the classes in relation to the IoU threshold is shown in the figure 5.20. The YOLO$_{\text{BVS}standard}$ model has a mean average precision (mAP) of 0.50 and a mean weighted F_1-measure of 0.93. When predicting the YOLO$_{\text{BVS}standard}$ model, we can conclude that it recognizes the *car* and *motorbike* class objects well. The mean average precision is lower compared to the YOLO$_{coco}$ model, but that precision does not take into account the number of instances in the validation set. The mean weighted F_1-measure is 0.93, which is a remarkable improvement compared to the YOLO$_{coco}$ model. The loss occurs in the prediction of the *truck* class. An example of the detection results is shown in the figure 5.21. It can be concluded that the YOLO$_{\text{BVS}standard}$ model gives better results than the YOLO$_{coco}$ model.

Model training using the transfer learning technique on the BVSstyle set

The YOLO model is used, which is trained on the BVSstyle dataset containing four classes and 6952 images. Further in the paper, the YOLO model trained on

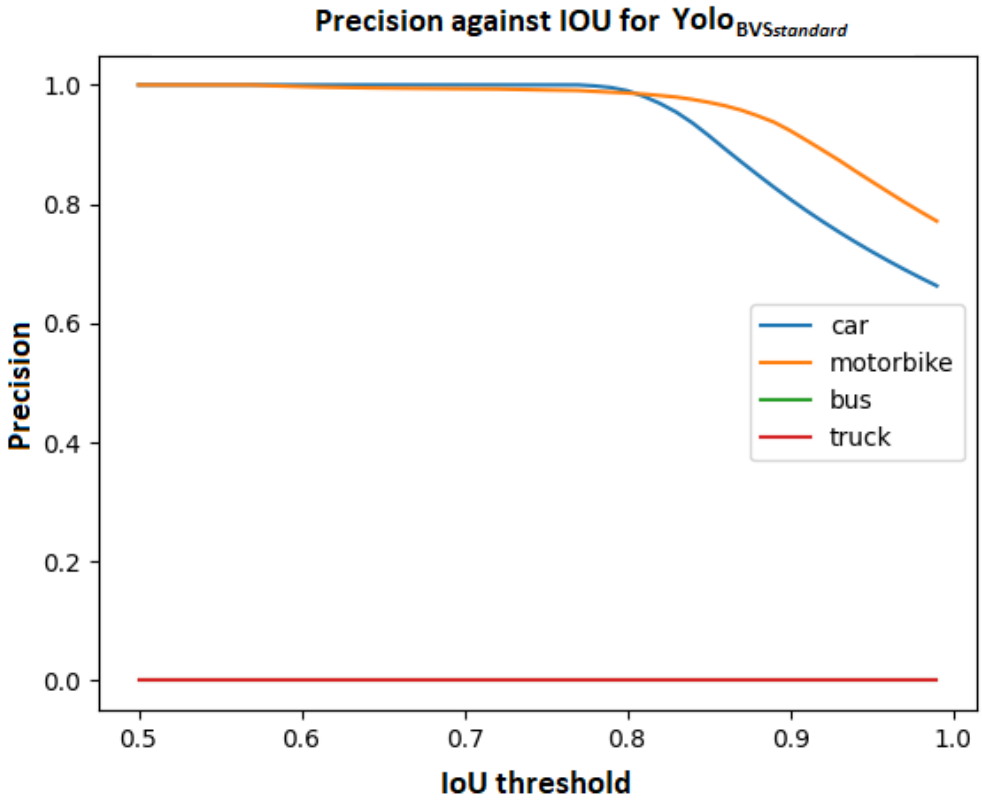

Figure 5.20: Precision relative to the IoU threshold [0.5,1) of the YOLO $_{\text{BVS}standard}$ model. For classes *bus* and *truck* the precision in all IoU values is 0.

the data set BVSstyle is denoted by YOLO $_{\text{BVS}style}$. The model loss function is shown in Fig. 5.22. It can be seen from the loss function that the model training stopped early after fourteen epochs. The loss function performs much worse than other models. During training, the model produced increasing errors. The model was trained on an NVIDIA Tesla P100-PCIE graphics card for two hours.

From the results of matrconfusion matrix (table 5.15) and model evaluation method (table 5.16) the following is concluded:

- class *car* model YOLO $_{\text{BVS}style}$ predicts for each instance. Eighty-nine out of a total of eighty-nine instances were predicted. The average precision for the *car* class is 1.0, while the mean F_1-measure is 0.96. These results indicate a perfect prediction of the YOLO $_{\text{BVS}style}$ model over the *car* class.

- class *motorbike* is predicted worse than class *car*. The YOLO $_{\text{BVS}style}$ model predicts seventy-eight out of a total of ninety-seven instances. Nineteen in-

Figure 5.21: Images from the validation set and predictions over them by the YOLO$_{\text{BVS}standard}$ model, class *car* (top), class *motorbike* (middle), class *truck* (bottom left) and the *bus* class (bottom right). An image without bounding boxes was not successfully classified by the YOLO$_{\text{BVS}standard}$ model.

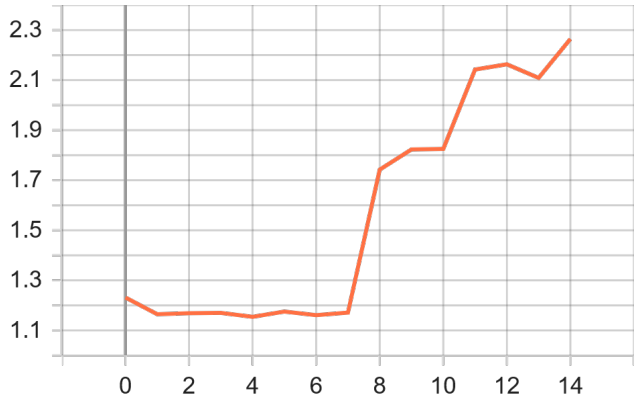

Figure 5.22: Loss function of the YOLO$_{BVS\textit{style}}$ model

Table 5.15: Confusion matrix for YOLO$_{BVS\textit{style}}$ with IoU threshold 0.5 and confidence threshold for recognized class 0.5

	car	motorbike	bus	truck	without detection
car	89	0	0	1	0
motorbike	0	78	0	0	0
bus	0	0	0	0	0
truck	0	0	0	0	0
without detection	0	19	1	0	0

Table 5.16: Model results YOLO$_{BVS\textit{style}}$

class	average precision (eq. 4.6)	mean F_1-measure (eq. 4.4)
emperor	1.00	0.96
motorbike	1.00	0.87
bus	0.00	0.00
truck	0.00	0.00
Mean average precision (eq. 4.7)	0.50	
Mean weighted F_1-measure (eq. 4.5)	0.90	

stances for the probability score have a value less than 0.5. The average precision is 1.0 and the mean F_1-measure is 0.87.

- class *truck* is incorrectly predicted as class *car*.
- class *bus* is not predicted by the YOLO$_{BVS\textit{style}}$ model.

The precision of the classes in relation to the IoU threshold is shown in the figure 5.23. The $\text{YOLO}_{\text{BVS}style}$ model has a mean average precision (mAP) of 0.50 and a mean weighted F_1-measure of 0.90. When predicting the $\text{YOLO}_{\text{BVS}style}$ model, we can conclude that it recognizes the *car* and *motorbike* class objects well. The mean average precision is lower compared to the YOLO_{coco} model, but that precision does not take into account the total number of instances in the validation set. The mean average precision is the same as that of the $\text{YOLO}_{\text{BVS}standard}$ model. The mean weighted F_1-measure is 0.90, which represents a remarkable improvement compared to the YOLO_{coco} model, but a deterioration compared to the $\text{YOLO}_{\text{BVS}standard}$ model. The loss, as with the $\text{YOLO}_{\text{BVS}standard}$ model, occurs in the prediction of the *truck* class. An example of the detection results is shown in the image 5.24. It can be concluded that the $\text{YOLO}_{\text{BVS}style}$ model gives better results than the YOLO_{coco} model, but slightly worse results than the $\text{YOLO}_{\text{BVS}standard}$ model.

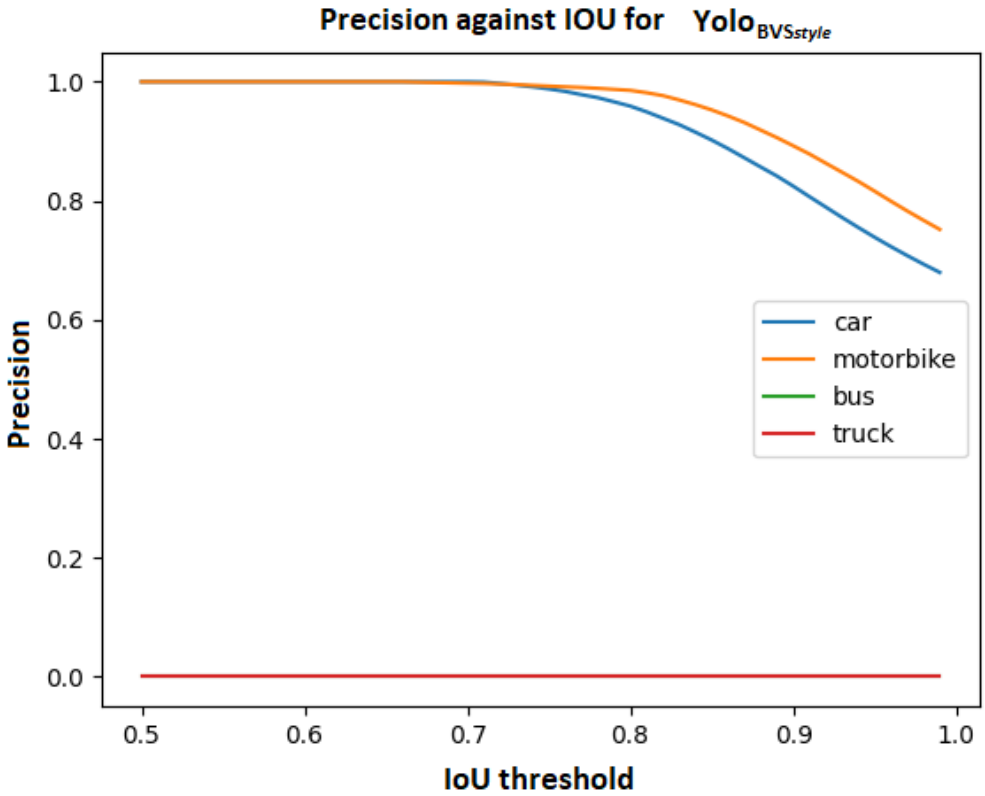

Figure 5.23: Precision in relation to the IoU threshold [0.5,1) of the $\text{YOLO}_{\text{BVS}style}$ model, for classes *bus* and *truck* the precision in all IoU values is 0

Figure 5.24: Images from the validation set and predictions over them by the YOLO$_{\text{BVS}style}$ model, class *car* (top), class *motorbike* (middle), class *truck* (bottom left) and the *bus* class (bottom right). An image without bounding boxes is not successfully classified by the YOLO$_{\text{BVS}style}$ model.

Chapter 6

Conclusion

In this paper, the problem of expanding the data set used for training models for object detection is solved. The dataset is augmented with images obtained by standard augmentations (translation, rotation, color change, etc.) and synthetically generated images by generative adversarial networks. In total, about five thousand images were added to the dataset. The thus extended data sets, BVSstandard and BVSstyle, were separately used to train the YOLO object detection model, while the VGG16 model was trained on the original BVS data set. In order to improve the results of the YOLO model, the transfer learning method was also used. The model weights used were trained on the COCO dataset. The YOLO model was trained on the BVS, BVSstandard and BVSstyle datasets.

The results obtained after training the YOLO model on the BVSstandard and BVSstyle datasets are better than the results obtained with the VGG16 model. The YOLO model trained on the BVS data set gives similar results to the VGG16 model. With the YOLO models trained on the BVSstandard and BVSstyle datasets, it can be concluded from the results that the precision has improved for the *car* and *motorbike* classes. For the YOLO model trained on the BVSstandard and BVSstyle datasets, the mean average precision dropped because the *truck* class is no longer recognized, which may be the cause of sharing features with the class *car*. All YOLO models perform worst when predicting the *bus* class.

Better results for object detection can be obtained by increasing the amount of data available to *StlyeGAN2* models and training them for a longer period of time. This expands the training set with higher quality and more diverse images generated by the *StlyeGAN2* model. Enhancement can also be achieved by using the union of the BVSstandard and BVSstyle datasets.

Based on the development of new technologies, it may be possible to generate synthetic data using entirely new methods. For example, the *DALLE-2* [30] and *MidJourney* [7] models are able to generate realistic images based on their textual descriptions. This approach, however, requires new data , i.e. textual descriptions of images.

Bibliography

[1] Yali Amit, Pedro Felzenszwalb, and Ross Girshick. Object detection. *Computer Vision: A Reference Guide*, pages 1–9, 2020.

[2] Dana H Ballard and Christopher M Brown. Computer vision. englewood cliffs. *J: Prentice Hall*, 1982.

[3] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36. JMLR Workshop and Conference Proceedings, 2012.

[4] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007.

[5] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[6] Lorenzo Brigato and Luca Iocchi. A close look at deep learning with small data. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2490–2497. IEEE, 2021.

[7] Jack Daniel, Max. midjourney creating images from text, 2022.

[8] DC Dowson and BV666017 Landau. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.

[9] Jie Feng, Xueliang Feng, Jiantong Chen, Xianghai Cao, Xiangrong Zhang, Licheng Jiao, and Tao Yu. Generative adversarial networks based on collaborative learning and attention mechanism for hyperspectral image classification, 2020.

[10] Rohith Gandhi. R-cnn, fast r-cnn, faster r-cnn, yolo - object detection algorithms, Jul 2018.

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[15] Nikola Grulovic. Vehicle detection data, 2022. Available at `https://drive.google.com/file/d/1EkrO28-iBCVWyPqy5mJXn8P8EoKQ6xUH/view?usp=sharing`.

[16] Nikola Grulovic. Vehicle detection github, 2022. Available at `https://github.com/Grula/vehicle-detection`.

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[18] Geoffrey M Hodgson. Microeconomics: Behavior, institutions, and evolution, samuel bowles, princeton university press and russell sage foundation, 2004, 584 pages. *Economics & Philosophy*, 22(1):166–171, 2006.

[19] Thomas Huang. Computer vision: Evolution and promise. 1996.

[20] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[21] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907, 2017.

[22] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[23] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[24] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[25] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[26] Anđelka Zečević Mladen Nikolić. *Mašinsko učenje*. Beograd, 2019.

[27] Anđelka Zečević Mladen Nikolić. *Naučno izračunavanje*. Beograd, 2019.

[28] Mae Mu. Orange fruit photo, 2019. [Online; accessed August 16, 2022].

[29] Deepthi Narayan, Srikanta Murthy, and G Hemantha Kumar. Image segmentation based on graph theoretical approach to improve the quality of image segmentation. *International Journal of Computer and Information Engineering*, 2(6):1803–1806, 2008.

[30] Mr D Murahari Reddy, Mr Sk Masthan Basha, Mr M Chinnaiahgari Hari, and Mr N Penchalaiah. Dall-e: Creating images from text. *UGC Care Group I Journal*, 8(14):71–75, 2021.

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[33] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[35] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[36] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Data-efficient gans with diffaugment. Technical report, 2020.

[37] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33:7559–7570, 2020.

[38] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[39] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

Biography of the author

Nikola Grulović (*Belgrade, Serbia, July 14, 1993*) graduated in Computer Science from the University of Belgrade. He graduated from the "Fifth Belgrade High School", majoring in natural sciences in 2012 in Belgrade. He entered the Faculty of Mathematics in Belgrade in 2013, majoring in Informatics and graduated in September 2018. After completing his academic studies, he enrolled in master's studies, and during the 2019-2020 academic year, he passed all the exams provided for in the plan and program of master's studies. In the period from November 2020 to April 2022, he worked as an intern at *AM Energia*, Sorocaba, Brazil. During his internship, he worked on improving existing systems and upgrading systems with new technologies.