TECHNICAL UNIVERSITY BRNO

BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGIES

FACULTY OF INFORMATION TECHNOLOGY

INSTITUTE OF COMPUTER GRAPHICS AND MULTIMEDIA

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

USE OF GAN-TYPE NETWORKS FOR DETECTION PRECISION AND TRAFFIC SIGN RECOGNITION

IMPROVING ACCURACY OF DETECTION AND CLASSIFICATION OF TRAFFIC SIGNS WITH GANS

BACHELOR THESIS

BACHELOR'S THESIS

AUTHOR OF THE WORK **MICHAEL GLOS**

AUTHOR

WORK MANAGER **doc. RNDr. PAVEL SMRŽ, Ph.D.**

SUPERVISOR

Brno 2021

Technical University Brno
Faculty of Information Technologies

Institute of Computer Graphics and Multimedia (UPGM) Academic year 2020/2021

Bachelor thesis assignment

Student: **Glos Michal**

Program: Information Technology

Name: **The use of GAN-type networks to refine the detection and recognition of traffic signs**

Improving Accuracy of Detection and Recognition of Traffic Signs with GANs

Category: Artificial intelligence Tasks:

1. Learn about object recognition methods and available implementations for easy configuration and training of neural networks.
2. Study the issue of using GAN-type networks for expanding data sets and increasing the robustness of detection and recognition.
3. Based on the knowledge gained, design and implement a system that will improve the detection and recognition of road signs from real images using GAN.
4. Evaluate the results of the system on a representative sample of data.
5. Create a brief poster presenting the work, its objectives and results.

Literature:

- according to the supervisor's recommendation For the award of credit for the first semester, the
- following is required: functional prototype of the solution

Detailed binding instructions for the preparation of the thesis see https://www.fit.vut.cz/study/theses/ Thesis supervisor: Smrž **Pavel, doc. RNDr., Ph.D.**

Institute head: ÿernocký Jan, doc. Dr. Ing.

Entry Date: November 1, 2020 Submission Date: May 12, 2021 Approval Date: May 7, 2021

Abstract The

aim of this work was to expand the data set for traffic sign detection. Generative neural networks of the PatchGAN and Wasserstein GAN types, combining DenseNet and U-Net architectures, were used for the solution. The models were designed to synthesize images of real-looking traffic signs from images of their standards. The SSD object detection model trained only on synthetic data achieved an accuracy of 59.6% mAP, which is 9.4% better than the reference model trained only on original data. In the case of training the model on a combination of synthetic and original data, the SSD model achieved an accuracy of up to 80.1% mAP.

Abstract The

goal of this thesis was to extend a dataset for traffic sign detection. The solution was based on generative neural networks PatchGAN and Wasserstein GAN of combined DenseNet and U-Net architecture. Those models were designed to synthesize real looking traffic signs from images of their norms. Model for object detection SSD, trained on synthetic data only, achieved mean average precision of 59.6%, which is an improvement of 9.4% over the model trained on the original data. SSD model trained on synthetic and original data combined achieved mean average precision of 80.1%.

Keywords SSD,

GAN, traffic sign detection, generative model, Pix2Pix, U-Net, Wasserstein GAN, PatchGAN.

Keywords

SSD, GAN, traffic sign detection, generative model, Pix2Pix, U-Net, Wasserstein GAN, PatchGAN.

Citation

GLOS, Michal. *The use of GAN-type networks to refine the detection and recognition of traffic signs.* Brno, 2021. Bachelor thesis. Brno University of Technology, Faculty of Information Technologies. Supervisor doc. RNDr. Pavel Smrž, Ph.D.

The use of GAN-type networks to refine the detection and recognition of traffic signs

Declaration I

declare that I prepared this bachelor's thesis independently under the supervision of doc.

RNDr. Pavla Smrž Ph.D. I have listed all literary sources, publications and other sources from which I drew.

. . . . . .

Michal Glos

May 11, 2021

Acknowledgments

I would like to thank my supervisor doc. RNDr. To Pavlov Smrž Ph.D. for valuable advice, factual comments and helpfulness during consultations and the preparation of the bachelor's thesis.

Content

Chapter 1

Introduction

Machine vision and artificial intelligence methods have been intensively studied not only in recent years. Newly discovered algorithms for object recognition, together with the growing computing capabilities of modern computers, have found application in various areas of industry and services. Their basis is often neural networks, which, however, often require a large amount of sufficiently high-quality data to function reliably. However, collecting data and annotating it is a very expensive activity, both financially and time-wise.

The goal of this work is to design and implement a system including generative neural networks, which would be able to automatically expand the data set intended for training models for object detection. By using such a system, it would be possible to train an accurate detector even with the use of a smaller data set, which could partially eliminate the expenses for its creation.

In this work, I focus primarily on the implementation and configuration of a model for the synthesis of annotated images of road signs from their standards defined by legislation. I gradually tuned the configuration of the generator based on the knowledge gained through experimentation in Chapter 6. Finally, I used the two generative models, producing the best visual quality images, to expand the non-public data set for the detection of traffic signs from the Slovak National Force. The detector, trained on a combination of real and synthetic data, achieved an accuracy of 80.1% mAP and after training only on synthetic data, 59.6% mAP. Both results show an improvement over the detector trained only on the original data, reaching an accuracy of 50.2% mAP.

In chapter 2 , I first describe modern neural network architectures for object recognition and the possibility of quantifying their accuracy. In the next chapter 3 , I address the architectures of generative GAN models and their use for image transformation. In chapter 5 , I describe the models used for both detection and data generation and the procedures for creating specific training data sets for both models. This chapter also summarizes the configuration options of the used models. In the final chapter 6 , different architectures and configurations of the generative model are experimented with in order to find the combination that generates the best quality synthetic data.

Chapter 2

Object recognition using convolutional neural networks

The concept *of object recognition* can refer either to *classification* - the task of determining the class of objects located in the image, or to *detection* - the task of correctly locating objects in the image, including their classification [31]. Most object recognition methods work in two steps.
First, characteristic features of the image are extracted, for example using convolutional neural networks, which are later classified into classes, for example using a *Support Vector Machine (SVM).* In recent years, many different approaches to the object recognition problem have been experimented with [25].
For feature extraction from images, methods based on *Haar [20], HOG (Histogram of Oriented Gradient) [5]* or *SIFT (Scale Invariant Feature Transform)* [23] were used . However, in 2012, at the *ImageNet Large Scale Visual Recognition Challenge* [31] in the category of object classification, the convolutional neural network *AlexNet* [18] clearly won , *see table* 2.1. This increased the popularity of convolutional neural networks for feature extraction to such an extent that most teams in the coming years of the competition were based on this architecture [32].

Model	**Top-1 Top-5**
Sparse coding [31] 47.1% 28.2% SIFT +	
FVs [33] 45.7% 25.7 % CNN **37.5% 17.0%**	

Table 2.1: Comparison of the best results achieved on the ILSRVC 2010 to 2012 test dataset [18, 31]. The **Top-1** column indicates the percentage of cases where the model did not predict the correct object class. **Top-5** indicates the percentage of cases when the correct class of the classified object does not occur among the 5 most probable estimates of the model [18].

2.1 Modern architectures of convolutional neural networks ur used to recognize objects

Because of such significant progress in classification accuracy using convolutional neural networks, researchers have sought to generalize this principle to the detection task as well. Several architectures were created, which in their performance far surpassed other, *state-of-the-art,* models at the time .
One of the first successful architectures was R-CNN [10], from which other successful models such as *Fast R-CNN* [9] and *Faster R-CNN* [29] were later derived. Other very successful methods are SSD *(Single Shot Detector)* [22] or YOLO *(You Only Look Once)* [28].

R-CNN

Region-Based Convolutional Neural Network or *Regional convolutional neural network,* abbreviated *R-CNN,* is one of the first methods successfully applying convolutional neural networks to the problem of object detection [10]. The model consists of a total of three components, *see Figure 2.1.* The first of them is an algorithm proposing the required number of regions with a probable occurrence of the object. In the original work, the *selective search* algorithm was used [36]. Another is a feature extractor in the form of a convolutional neural network, which is gradually applied to all proposed regions. The last one is the classifier, which classifies the extracted features of all proposed regions into classes. The disadvantage of this model is that each of the proposed regions must be classified separately. The original work [10] proposes 2000 regions during model validation, which makes it practically impossible to use R-CNN in real time.

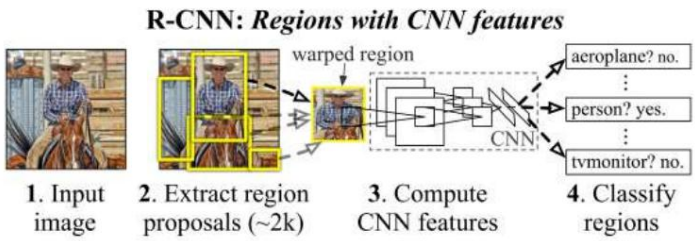


Figure 2.1: Graphic representation of all steps of the object detection process by the *R-CNN model. Adapted from [10]*

Fast R-CNN

After the great success of *R-CNN, the Fast R-CNN* model [9] was proposed in a publication by the same author in 2015, which mainly addresses speed problems, but also achieves higher accuracy.
Compared to *R-CNN,* the training process is 9 times faster and the evaluation process up to 213 times faster [9]. Its main advantage is the fact that it is designed as one monolithic model, *see Figure 2.2,* unlike *R-CNN,* which is composed of three separate components. Instead of processing each region with a convolutional feature extractor separately, Fast R-CNN calculates the features for the entire input image at once, forming a feature map. From this map, the proposed regions are then selected, which are transformed into vectors of constant length, from which information about the class of the object and the position of its *bounding box* is subsequently obtained by applying fully connected layers and an activation layer.

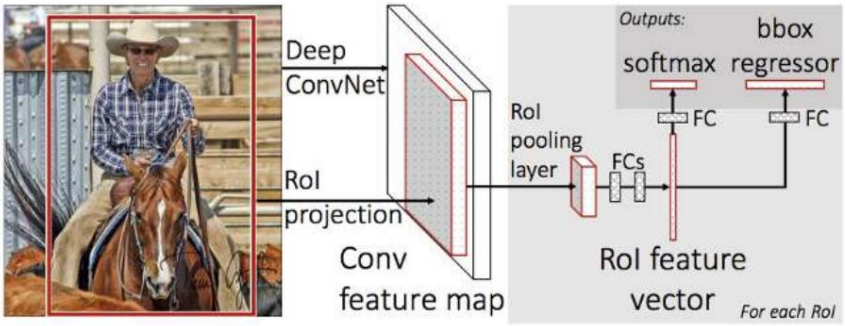


Figure 2.2: Schematic of the *Fast R-CNN model.* The input is an image to which a feature extractor in the form of a deep convolutional neural network is applied. The features inside the proposed regions of different sizes are transformed into vectors of the same size using the *RoI pooling* layer. These are then processed in two branches. In the first branch, several fully connected layers and a *softmax* activation function are applied to the vector to determine the class of the object.
In the second branch, several fully connected layers are also applied to the vector, but these lead to *the bounding box regressor,* thereby specifying the position of the bounding box of the detected object.
Taken from [9]

Faster R-CNN

The method of designing regions using the *Selective Search* algorithm is very slow, and its implementation on a CPU generates regions for each image in 2 seconds [29], which revealed the bottleneck of previous models. Therefore, the *Faster R-CNN* method uses a separate *RPN neural network (Region Proposal Network)* to propose regions , achieving an order of magnitude faster than the *Selective Search algorithm,* and is considered the first method capable of detecting objects in real time [22]. The *Fast R-CNN* method is an order of magnitude faster relative to *R-CNN ,* but it is still not sufficient for real-time object detection [9].

SSD *(Single Shot Multibox Detector)*

The *SSD* method was created with great emphasis on the speed of the entire process. It can be divided into two parts. First, it uses the *VGG16* network to extract a feature map from the input image, to which convolutional layers are subsequently applied, thereby obtaining feature maps of multiple sizes [22], *see Figure 2.3.* Several default bounding boxes of different sizes and aspect ratios are designed for each cell contained in these maps. Their absolute dimensions are the same in all layers of feature maps, therefore, in maps with a larger number of cells, objects occupying a smaller part of the image are detected, and as the number of cells in the maps decreases, larger objects are detected [22], see Figure *2.4 .* The size of the smallest and largest feature maps can be configured in the model [22]. It is advantageous to set these values according to the data set used, so that computing resources are not wasted through unnecessary attempts to detect significantly larger or smaller objects. A 3x3 convolutional filter is applied to the feature map in each default bounding box , the output of which is a vector containing information about the class of the object and the offset of its bounding box [22].

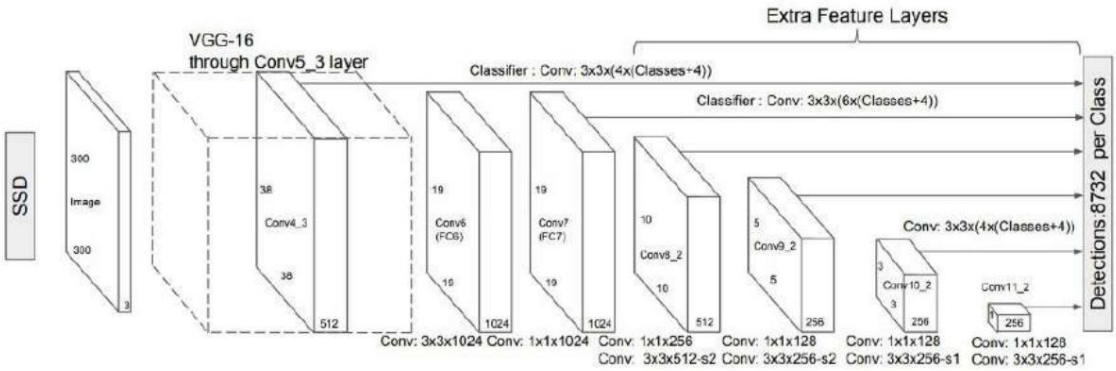


Figure 2.3: On the *SSD* architecture diagram, you can see how feature maps of different sizes are obtained by successive application of convolutional layers. Adapted from [22]

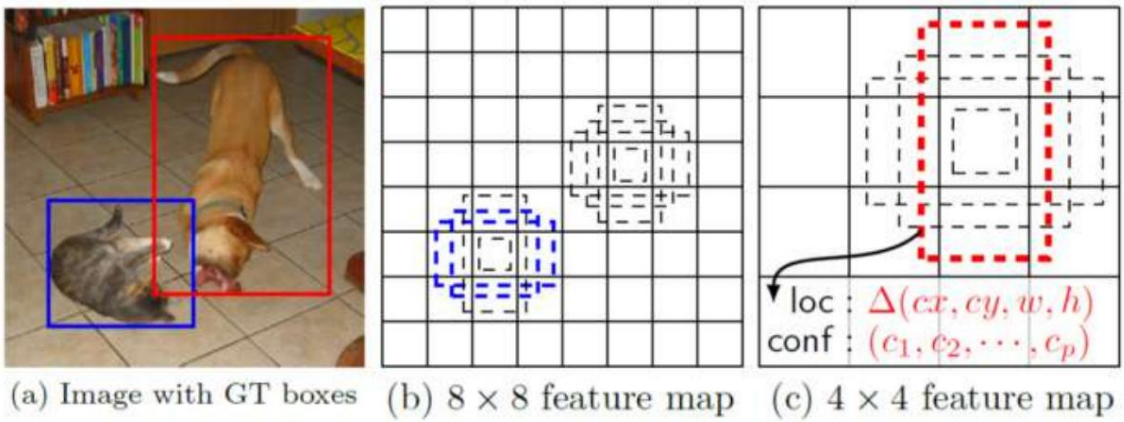


Figure 2.4: Searching all feature maps using bounding boxes of the same absolute dimensions allows searching for objects of different sizes in each of the layers. Adapted from [22]

2.2 Model evaluation metrics

Competitions and challenges in machine vision advance the capabilities of object recognition almost every year. In order to be able to compare the accuracy of individual, not only competitive, models, we need to quantify the quality of their output. Several different methods can be used to do this.
I mention the most popular ones in this section. The accuracy of models used exclusively for classification can be quantified relatively easily, because their output is only two values - the class of the object and the level of confidence of the model in its correctness. However, the output of a model designed to detect objects in an image is a bit more complex. It consists of a list of bounding boxes, object classes, and a list of values corresponding to the model's confidence levels in a given prediction. Bounding box is a "frame" bounding the detected object and its position is often given as the position of the point in its upper left and lower right corner in the format ( ). But it is not a rule, for example the YOLO algorithm [28] defines the bounding box by the position of the point in its center and its dimensions [25]. More advanced metrics and their combinations are already used in competitions and publications to quantify the accuracy of detector output [26, 8, 31], *see table 2.2.*

The most used of them are AP *(average precision)* and methods derived from it, such as mAP *(mean average precision)* [25], which I discuss in more detail in section 2.2.

Method	Data set	Metrics
Fast R-CNN [9]	PASCAL VOC [8]	, (= .50) (=
Faster R-CNN [29]	PASCAL VOC [8]	, .50) @[.5 :
Faster R-CNN [29]	COCO [21]	.05 : .95], @.50 (= .50) (=
R-CNN [10]	PASCAL VOC [8]	, .50) (= .50) (=
R-FCN [4]	PASCAL VOC [8]	.50) (= .50)
SSD [22]	PASCAL VOC [8]	
SSD [22]	ImageNet [32]	
YOLO v1 [28]	PASCAL VOC [8]	,

Table 2.2: Popular detector architectures and metrics quantifying their accuracy on polar datasets. *Data taken from [26]*

Basic model evaluation metrics

The most basic options for quantifying the detector output are on:

- True positive (TP) - Correct detection and localization of the object
- False positive (FP) - Detection of a non-existent object, or inaccurate object localization existing
- False negative (FN) - An existing object was not correctly detected

The last option, True negative (TN), cannot be applied to the problem of object detection, since each scene contains an infinite number of bounding boxes in which no object is found [26].

In order to apply such a division at all, it is necessary to define in which case the model has located the object in the scene correctly and when it has not. For this, the value *IoU (Intersection over Union) is most often used,* expressing the ratio of the content of the intersection of the detected bounding box with the real bounding box, for the content of their union, *see equation* 2.1 *and figure* 2.5.

= ÿ(ÿ) ÿ(ÿ) (2.1)

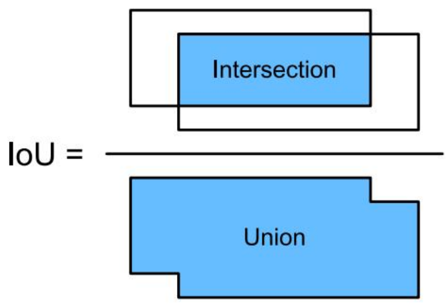


Figure 2.5: Graphical representation of the IoU calculation. *Taken from www .roelpeters .be*

Since the union of two surfaces will always have the same or greater content than their intersection, the value of IoU varies in the interval ÿÿ 0, 1 ÿÿ . If a threshold is defined, falling in the same interval, the detection can be considered successful if ÿ and as unsuccessful, < [25]. Another two very important concepts are *precision* and *recall*. *Precision* can be defined as the ability of the model to detect only relevant objects. Its value is the ratio of the number of correctly detected objects to the number of all detections of the model [25], *see equation* 2.2. *Recall* can be defined as the ability of the model to detect all existing objects [25]. Its value is calculated as the ratio of the number of all correctly detected objects to the number of all object annotations existing in the data set, *see equation* 2.3 [26].

The sum of correct detections and detections of non-existent objects is equal to the number of all detections made by the given model. The sum of correct and minute detections corresponds to the number of all annotations in the given data set. The following relationships can be used to calculate *Precision* and *Recall* .

= + = (2.2)

= + = (2.3)

Advanced model evaluation metrics based on *precision* and *recall*

As I mentioned above, *see 2.2,* the output of the detector consists of bounding boxes, classes, and confidence levels of the model. The only part of this output that is not accounted for in the above methods is the confidence level of the model. This can be included in the *precision* and *recall* calculations , which gives us additional information about the behavior of the model. This can be achieved by specifying a threshold in the interval ÿÿ 0, 1 ÿÿ . By comparing the threshold with the degree of confidence of the model in the correctness of the detection, we will find out if the detection will be included in the calculations or if it will be discarded. *Precision* and *recall* can be calculated depending on the threshold according to relations 2.5 and 2.4. These equations are a slight modification of equations 2.2 and 2.3. The values of ( ), ( ), and ( ) can be expressed as the number of detections in a given group with a confidence level of the model greater than the threshold and are non-increasing functions. The number of undetected objects can also be expressed as a function of the threshold , and the sum of the number of detections ( ) with a confidence level less than or equal to the threshold . This function does not decrease depending on the threshold. The number of all annotations in the dataset is threshold value independent and thus has the same meaning as in Equations 2.3 and 2.2.

$$( ) = \frac{( )}{( ) + ( )} = \frac{( )}{( )} \tag{2.4}$$

$$( ) = \frac{( )}{( ) + ( )} = \frac{( )}{} \tag{2.5}$$

In practice, a good detector should find all annotated objects in the test dataset. This means that the value would be very low, thus the *recall* value would be high. Furthermore, it should only identify relevant objects, so the value would be very close to zero, so the *precision* value would also be high. Therefore, one can be considered a good detector for which, as the threshold value decreases, *the precision* remains at a high value and *the Recall* increases [26]. By plotting the dependence of *precision* on *recall ,* we obtain the so-called *PR* curve, which is characterized by its 'zig-zag' shape [26], see figure 2.6. Although plotting *the PR* curve gives us a lot of information about the model's behavior, it is much more convenient to quantify their accuracy with a single value to compare their performance. Therefore, the *mAP (Mean Average Precision)* method is often used , where the quality of the model is expressed using the mean *AP (Ave rage precision)* for each class in the data set, *see equation 2.6.*

$$= \ddot{y}\ddot{y} \quad \frac{( )}{| |} \tag{2.6}$$

Value refers to the set of all classes in dataset a ( ) *average precision* is calculated only for the set of detections of class objects. *AP* can be calculated as the definite integral of the graph under *the PR* curve in the interval ÿÿ 0, 1 ÿÿ . Due to the peculiar 'zig-zag' course of *the PR* curve, the calculation is often simplified by interpolating this function in points. The resulting function ( ) is monotonic and can be expressed using relation 2.7.

$$( ) = \max_{:} ( ) \ddot{y} \{ ( )\} \tag{2.7}$$

Value ( ) is equal to the maximum of the function ( ) for which ( ) ÿ .

In practice, the value of *average precision* is approximated by calculating *the Riemann integral* in bo breath from a set ( ) of cardinality , whose elements are defined by the relation

$$( ) = \frac{}{ÿ\ 1}, \quad = 1, 2, ..., \quad \cdot \tag{2.8}$$

The elements of this set are uniformly distributed over the interval ÿÿ 0, 1 ÿÿ and contain the values 0 and 1. The *average precision* approximation itself can therefore be written by expression 2.9.

$$= \frac{1}{} \sum_{=1} ÿÿ \quad (\quad ( )) \tag{2.9}$$

AP interpolation at 11 points is used when evaluating results on the *Pascal* dataset [8] and at 101 points when evaluating results on the *COCO* dataset [21].

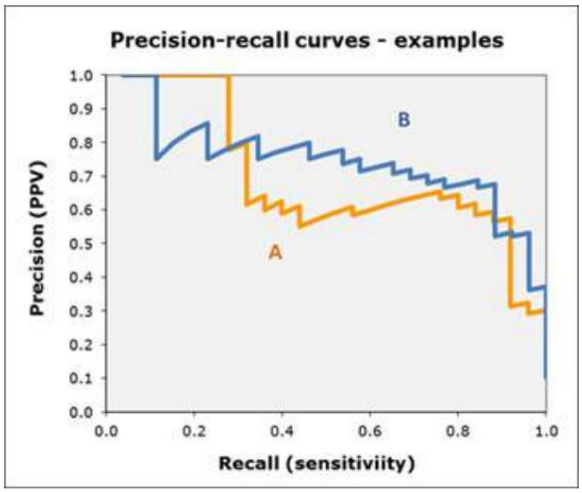


Figure 2.6: The graph shows two *PR* curves with a typical "zig-zag" pattern. Image taken from acutecaretesting.org.

Chapter 3

Neural networks of the GAN type

One way to improve the performance of neural networks is to augment their training dataset using a generative model. One of the groups of generative models are the so-called *GANs (Generative Adverserial Network),* which I deal with in this chapter. GANs automatically recognize patterns in input data. Assuming the convergence of their training process, they are able to generate synthetic data, almost unrecognizable compared to real data, based on the probability distribution of the training data [12]. The GAN architecture includes a total of 2 neural networks and is shown in *Figure 3.1.* A neural network that, based on the input defined by the architecture used, generates synthetic data similar to the original data set is called *a generator.* However, it needs another neural network - *a discriminator - for its training.* Its input can be a sample from the original data set or the output *of a generator ,* and its task is to determine whether the input data is real or not. Based on the output *of the discriminator,* the generator weights are adjusted .

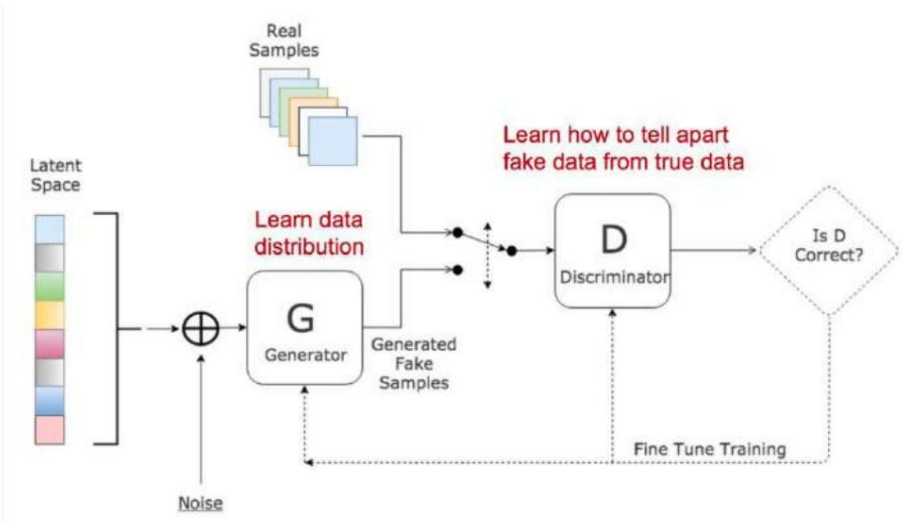


Figure 3.1: General diagram of a GAN. *The generator* generates samples from random noise. *The discriminator* is trained on real and generated data and its weights are adjusted independently of the entire model. The output of the discriminator is then used together with the loss function *of the generator* to calculate new weights for *the generator.*

GANs can be described simply by analogy with banknote counterfeiters. They try to fake them so well that they remain undetected. The discriminator plays the role of a policeman trying to detect fakes. The rivalry between the police officer and the counterfeiter leads both parties to perfect their methods. After a sufficiently long period of continuous improvement of the forger's and the policeman's techniques, the forger learns to create banknotes that are absolutely identical to the real ones [12].

GAN-type networks have undergone rapid development since their first introduction, *see Figure 3.2.* The DCGAN architecture, first presented in 2016 [27], uses deep convolutional neural networks to implement *both the generator* and *the discriminator, see Figure 3.3 .* The results of this method are so satisfactory that most modern GAN network architectures are at least partially derived from DCGAN [11].

Figure 3.2: Demonstration of the rapid development of GAN networks. Each image is the output *of the state of the art* generator in that year. *Adapted from [27]*

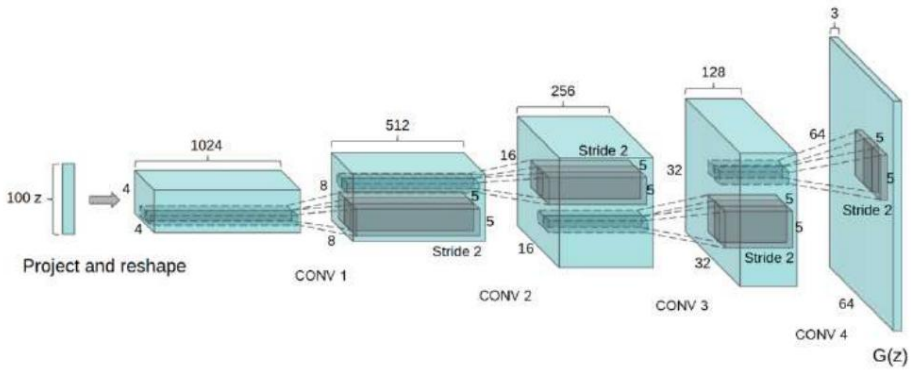


Figure 3.3: DCGAN generative model architecture for modeling synthetic images from the LSUN dataset [41]. A 100 unit long vector sampled from a uniform distribution is mapped into a spatially small convolutional representation with many feature layers. This is followed by a series of four fractional step convolutional layers that convert the low-dimensional input up to a 64x64 image . No fully connected or pooling layers are used in the model. *Adapted from [27]*

3.1 Training the generative model

GAN-type networks are known for their instability during the training process and often generate completely meaningless data [27]. The original publication presenting GANs approaches their training as a two-player game using the minmax method, where *both the generator* and *the discriminator* are trained at the same time. This minmax game can be written by equation 3.3, where ( ) represents *the discriminator* function , which takes the value 1 if the discriminator thinks it has real data at the input, and takes the value 0 for generated data. To train the *discriminator ,* it is necessary to maximize the probability that *the discriminator* correctly recognizes real data from false ones, *see equation 3.1.* ( ) represents the output of a dependency *generator* trying to create real-looking data minimize the probability that the discriminator will distinguish it from the real data, see equation 3.2. It is necessary to real ones, *see equation 3.2.* Cross entropy is used to calculate probabilities [12].

$$\max ( ) = E [\log ( )] + E [\log(1 \ddot{y} (( ))] \quad (3.1)$$

$$\min ( ) = E [\log(1 \ddot{y} (( ))] \quad (3.2)$$

$$\min \max (,) = E [\log ( )] + E [\log(1 \ddot{y} (( ))] \quad (3.3)$$

In practice, however, it has been shown that these functions may not provide the generator with a strong enough gradient for effective learning in the early learning phase. At the beginning of the training process, when the generator does not produce data similar to real ones, the discriminator can distinguish the data with great accuracy, making the training ineffective [12]. One possible solution is to use a non-saturating loss function, where instead of minimizing the probability that the discriminator will detect false data, we will maximize the probability that the discriminator will classify false data as real [12], *see equation 3.4*

$$\max ( ) = E [\log((( ))] \quad (3.4)$$

Wasserstein GAN

Although GANs are capable of producing high-quality synthetic data, their training is often unstable and several failure modes may occur [2], despite following the recommendations presented later in this section . One such mode is the so-called *Mode Collapse,* when data is generated only from some parts of the probability distribution of the training data set [35]. The instability problems of GANs have still not been fully resolved and are the subject of current research. However, several procedures have been proposed to at least partially eliminate these problems [12, 35]. One of them is the use of *Wasserstein GAN (WGAN)* [2], which is an extension to the original model [12]. *WGAN* achieves better stability during training and provides a loss function that correlates with the quality of generated data [2]. During training, *WGAN* tries to minimize the distance between the data provided for training and the generated data, which it defines as *the Earth-Mover* or *wasserstein* distance. This can be formulated as the lowest cost of moving mass when converting from a probabilistic data distribution to a probabilistic data distribution [2]. The value of this distance is approximated by the so-called *critic,* which replaces the function of the discriminator in the *WGAN* model . Discriminator

very quickly learns to classify the input data into real and fake, thereby ceasing to provide reliable information about the gradient for adjusting the generator weights [12]. However, *the critic* provides a loss function that is continuous and differentiable at almost all points. Therefore, during training, *the critic* provides information about the gradient with ever-improving quality [2], which increases the stability of the generative model and decreases its sensitivity to the configuration of hyperparameters [2]. During the model training process, *the critic* weights are adjusted more often than *the generator weights,* in the original publication [2] the authors trained *the critic* five times more often than *the generator.*

Rules for stable training of GANs

DCGAN [27] also published a list of recommendations for stable training that can be applied to most GAN architectures.

- Replace all pooling layers with convolutional layers with a larger step for both the discriminator and the generator and use convolutional layers with a fractional step to increase the resolution in the generator
- Use batch normalization in both the generator and the discriminator
- Replace fully connected layers with deeper architectures
- Use *the ReLu* activation function for all layers of the generator, except the last one will use the *Tanh* activation function
- Use *the LeakyReLu* activation function for all *discriminator* layers

3.2 Data synthesis and transformation using generative mo

cannon

The models described above are able to generate visually very high-quality images [27]. However, their functionality is limited by the fact that the generator accepts only random noise as input. This is the reason that synthetic data is generated randomly based on the distribution of training data and its properties cannot be well controlled during generation [24]. Some control over the properties of the generated images can be obtained by conditioning the input of the discriminator and generator with additional information about the object, most often its class [24]. The input of the discriminator is extended by information related to the assessed image, and the generator by information that defines the property of the generated image. Such a model is called *Conditional GAN* [24] and is shown in *Figure 3.4.*

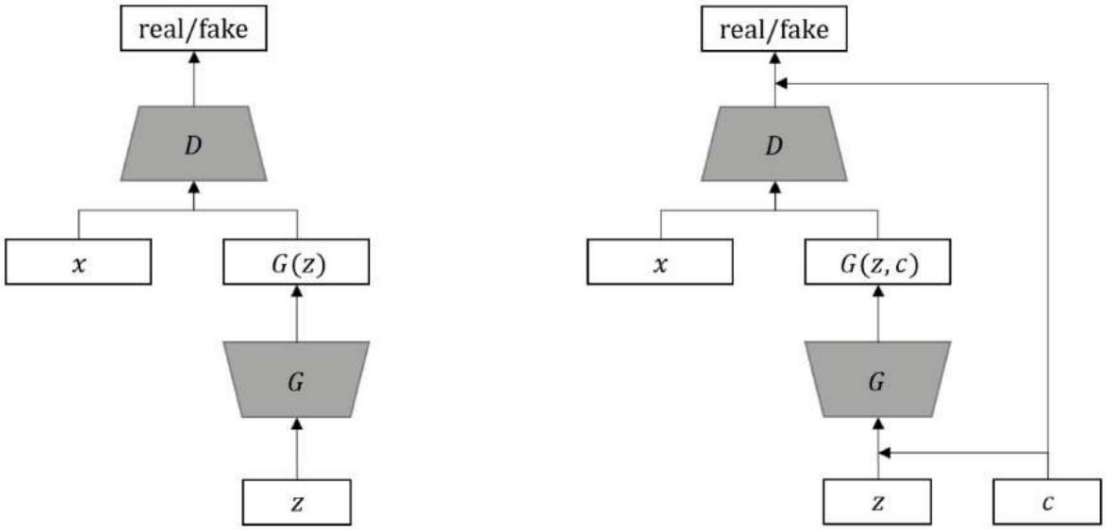


Figure 3.4: On the left is a diagram of an unconditional GAN-type network model, composed of a generator with input in the form of noise and a discriminator with input in the form of original or generated data. On the right, you can see the diagram of the *Conditional GAN* model and its extension of the generator and discriminator with additional information. *Adapted from [19].*

Pix2Pix

The *Pix2Pix* model is an extension of the *Conditional GAN* method and serves to translate an image into an image, where an image of one form is transformed into an image of another form, while preserving its semantic meaning [16]. For example, if the generator is given an image taken in the rain, its output can be an estimate of what the same scene would look like if it wasn't raining. Traditionally, each of the image-to-image translation problems was solved by specialized methods [16, 7], but the *Pix2Pix* model is designed with the idea of generalization and can be applied to most of these problems, for example, transforming satellite photos into maps, generating realistic product photos from their sketches or coloring of black and white photographs [16]. By its very nature, *Pix2Pix* needs paired data for training, a pattern for the transformation and its target. Like other GANs, it is made up of two competing neural networks.

In the original publication [16], the U-Net model [30] is used as the generator and Patch-GAN [16] as the discriminator.

Patch GAN

Ordinary GANs work with a discriminator that quantifies the reality of the entire image into a single value [12, 27]. However, this method can generate blurry and otherwise low-quality images when used at higher resolutions [16]. *Patch-GAN* solves this problem to some extent by evaluating the reality of individual parts of the scene [16]. The reality of the input is examined by its parts and is quantified by a matrix of values corresponding to the reality of these parts. The dimensions of the evaluated slices are defined by the architecture of the discriminator, and the best results when transforming images of size *256x256* were achieved by evaluating the realism of images in parts of size *70x70* [16].

U-Net

U-Net is a modern neural network architecture that transforms input images using convolutional filters [30]. Its structure is based on the *Encoder-Decoder* 3.5 architecture, but it also uses *skip connections,* the connection of convolutional layers of the same size between *the encoder* and *the decoder.* Compared to previous methods , *U-net* can produce high-quality outputs even from a smaller number of examples and depends mainly on large data augmentation [30].

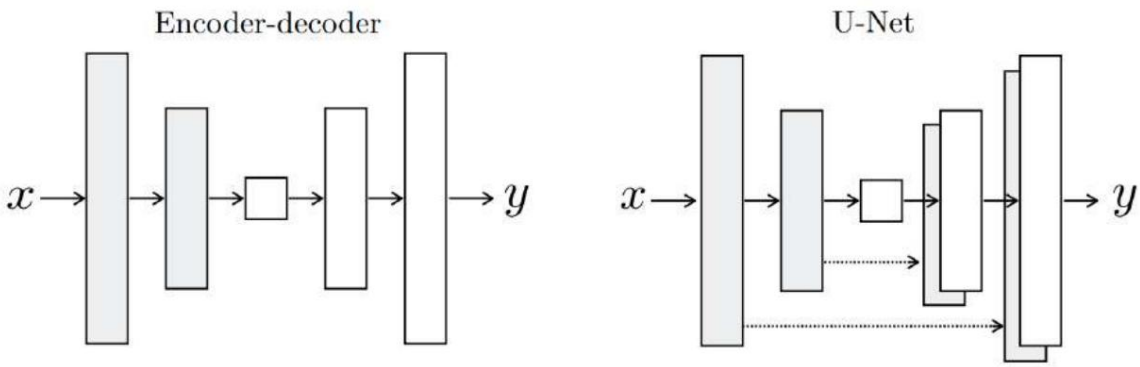


Figure 3.5: Comparison of *Encoder-Decoder* and *U-Net architectures.* The parts of the network, colored gray, are called the encoder. At their end is the narrowest point of the model, the so-called *bottleneck .* The second part of the network with white blocks, decoders, follows. Functional blocks of individual parts consist of convolution filters, batch normalization and *ReLu* [16]. Convolutional filters use step > 1 in the encoder and step = 1/ in the decoder. In addition, *U-net* uses *skip connections,* which connect encoder outputs with decoder inputs of the same size. *Adapted from [16].*

RU-Net

The *ResNet* architecture [13] , compared to the direct architecture of convolutional neural networks for feature extraction, achieves better results and more stable training [13]. *ResNet* networks are made of residual blocks, *see Figure 3.6.* The residual block consists of several convolution and activation layers and batch normalization. However, the input of the residual block is added to the output of one of its last layers. RU-Net [42] combines the principles of the U-Net and ResNet architectures to achieve better results in semantic scene segmentation [42]. Instead of encoders, RU-Net uses a residual block followed by a convolutional layer with a larger step, and instead of decoders a residual block followed by a convolutional layer with a fractional step.

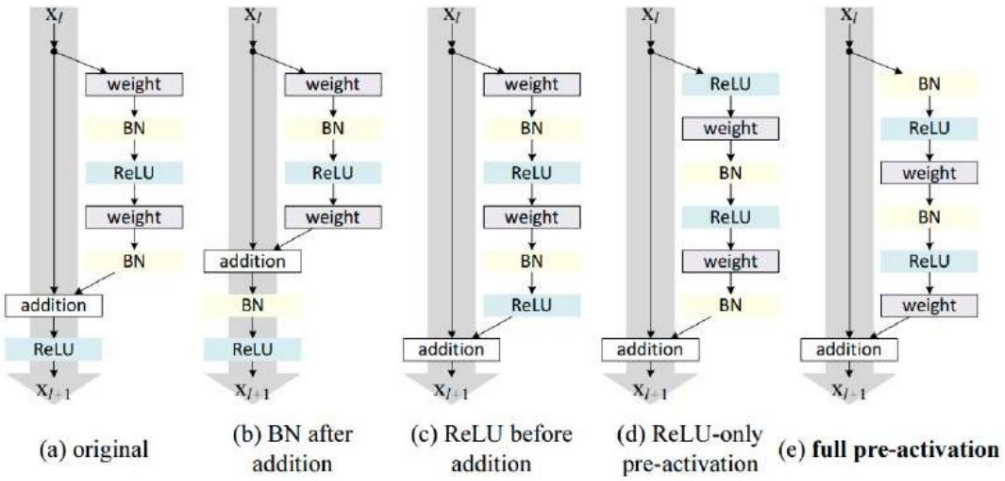


Figure 3.6: Several possible residual block architectures consisting of the same components but in a different order. These architectures were tested on the CIFAR 10 dataset [17], and the best result was achieved by the architecture in Figure e, which adds the block's input to its output [14]. The second most successful was the architecture used in the original ResNet publication [13, 14]. *Adapted from [14].*

Dense RU-Net

Dense RU-Net is a similar extension to RU-Net, but instead of residual blocks, DenseNet [15] uses residual blocks. These consist of several smaller sub-blocks, containing a convolutional layer, a batch normalization and an activation layer. As with residual blocks, the input of a DenseNet block is summed with its output [39]. The input of each of the sub-blocks is created by connecting the input of the entire block with the outputs of all previous sub-blocks, *see Figure 3.7*

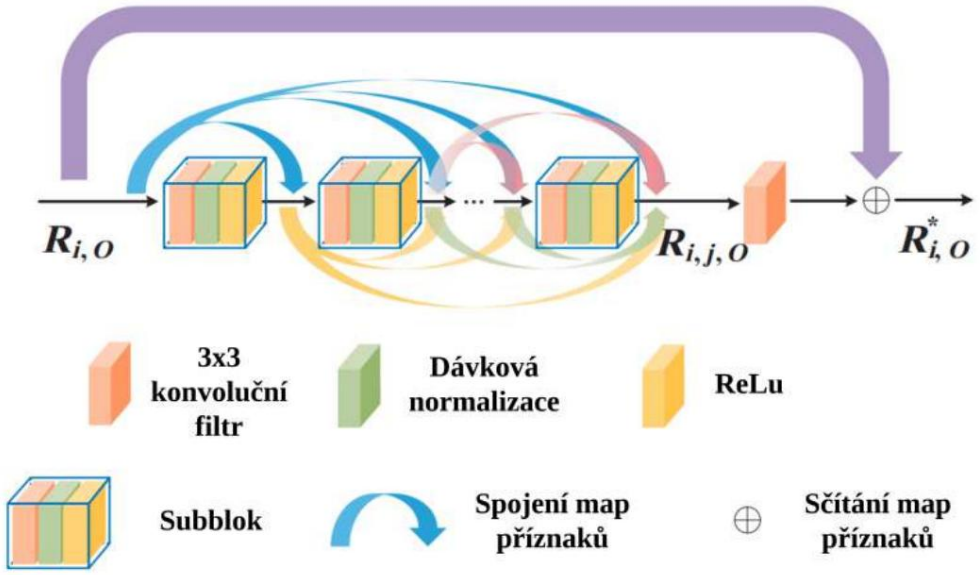


Figure 3.7: DenseNet block diagram. Compared to the classic one, it differs in the structure of how many sub-blocks are made up of them. Each sub-block input is then a combination of the outputs of all previous sub-blocks and the input of the entire block. The number of sub blocks can affect the quality of the generated images. *Adapted from [39].*

Chapter 4

Possibilities of using a generative model to expand datasets for traffic sign detection

Traffic sign detection falls under the set of computer vision tasks. The image of the real situation is captured by the camera and processed in such a way that it detects and classifies traffic signs in the captured scene. Road signs play an important role in navigating the roads. They define the maximum permitted speed, determine or prohibit the direction of travel, warn of possible dangers or adjust the right of way for vehicles.

4.1 Application of traffic sign detection systems

The first car with a traffic sign detection function was the Opel Insignia in 2008 [34]. However, he could only read the signs defining the maximum permitted speed and warn the driver if he exceeded it. With the increasing accuracy of object detection methods [31, 8] and the performance of smaller and smaller chips and sensors, the degree of applicability of such systems is increasing not only in the automotive industry. However, these systems no longer serve only to provide information to the driver, who acts on the basis of his own information, but the detection systems serve only as a secondary source of information. Today's modern systems are capable of very accurate and consistent recognition of traffic signs [1] and provide information to superior systems that control the vehicle autonomously, so the further operation of the autonomous system directly depends on the accuracy of the information provided. If such a system fails in at least one of the tasks of understanding the environment (path detection, obstacle detection, and traffic sign detection and classification) [6], the information about the environment is not completely valid and an accident may occur [38]. Despite this fact, however, by 2021 several cars with the option of fully autonomous control from Waymo , *Honda* or *Tesla are already legally driving on the roads.*

4.2 Problems in traffic sign detection

There are often large differences in the appearance of common objects, such as cars or people, even between instances of objects within the same class. However, road signs are defined by the legislation of the country and should not vary much within it. Most of the marks are in relatively good condition and do not differ much from their legally defined standards. Such brands are easy to detect because they use rich, bold colors. However, there are quite a few signs on the road, which are more difficult to recognize for several possible reasons, *see Figure 4.1.* The ability of the detector to make a correct decision in challenging situations can be supported by more frequent selection of such examples during training, or by expanding the dataset with challenging cases using a generative model. The system can encounter a plethora of unique situations on the road. Some of them are captured in datasets, but it is impossible to capture them all and in sufficient quantity. This creates a set of situations that, if the model encounters them in operation, may not be evaluated correctly, because the situation is unusual for the model. This can be, for example, a special deformation of a road sign, which can lead to its misclassification, or the model may not detect it at all. With the help of a generative model, these situations can then be synthesized, which means that the model can be well generalized to such cases as well. This can lead to an improvement in the consistency of the detecting model, and even to more robust and secure systems [3]. Companies developing fully autonomous cars rely heavily on data synthesis. Advanced generative models can produce many times more synthetic data compared to original data, in high quality [40]. In addition, the generation of this data is controlled to capture situations relatively unknown to the system, thereby significantly improving it.

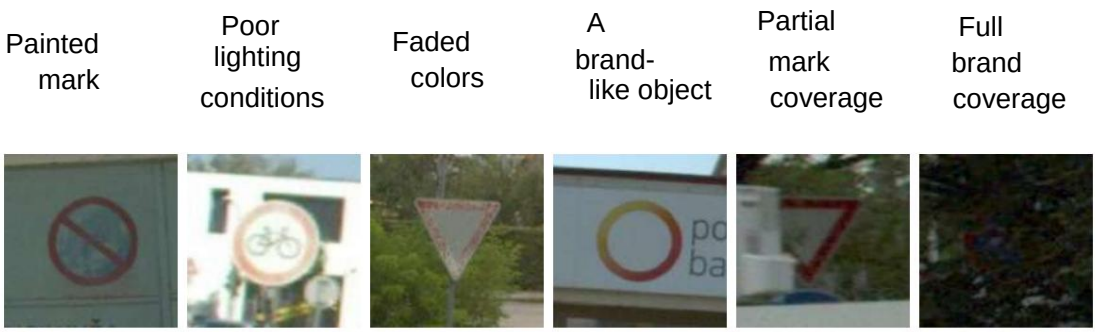


Figure 4.1: The figures show challenging cases for traffic sign detection models. Using a generative model, these difficult cases can be synthesized, thereby improving the detection of deformed or otherwise difficult to detect marks.

4.3 Brand differences across countries

Geographical localization plays a big role in traffic sign detection. Historically, the legal mark has differed in almost all countries, but for a long time there have been international efforts to unify this system in order to increase security. The largest such effort is *the Viennese one*

convention on traffic signs and signals1 , signed by 68 states as of August 2016, most of which have ratified this convention, *see Figure 4.2.*

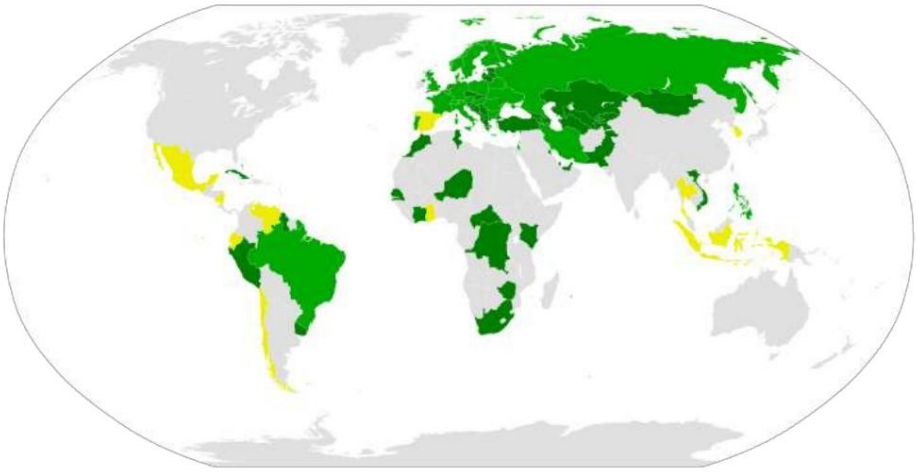


Figure 4.2: World map with countries using the standards defined in the Vienna Convention highlighted in color. States in light green have ratified the convention, states in dark green have signed it and are using it in practice, and states in yellow are using it without signing it. *Taken from www .wikipedia .org*

The fact that road signs are standardized will lead to greater portability of autonomous car driving systems between different states. Until then, due to the variety of traffic signs, datasets from local roads are often needed to train the detector in different countries, *see Figure 4.3.* The existence of the generative model proposed in Chapter 5 would eliminate this need by creating an artificial data set based only on road sign standards defined by the given country and the default data set.

Modern object detection model architectures are capable of highly reliable detection of traffic signs on public datasets [1]. Highly visible brands whose instances are more numerous in the training data set are almost perfectly detected by the models. However, the models often miss detection if there are few examples of a given sign class in the training data set, or if the given road sign is visually different, for example due to deformation or lighting conditions. The generative model proposed in Chapter 5 can synthesize high-quality images containing objects of rarely occurring classes, thus compensating for their deficit during learning.

1Available at: https://unece.org/DAM/trans/conventn/Conv_road_signs_2006v_EN.pdf

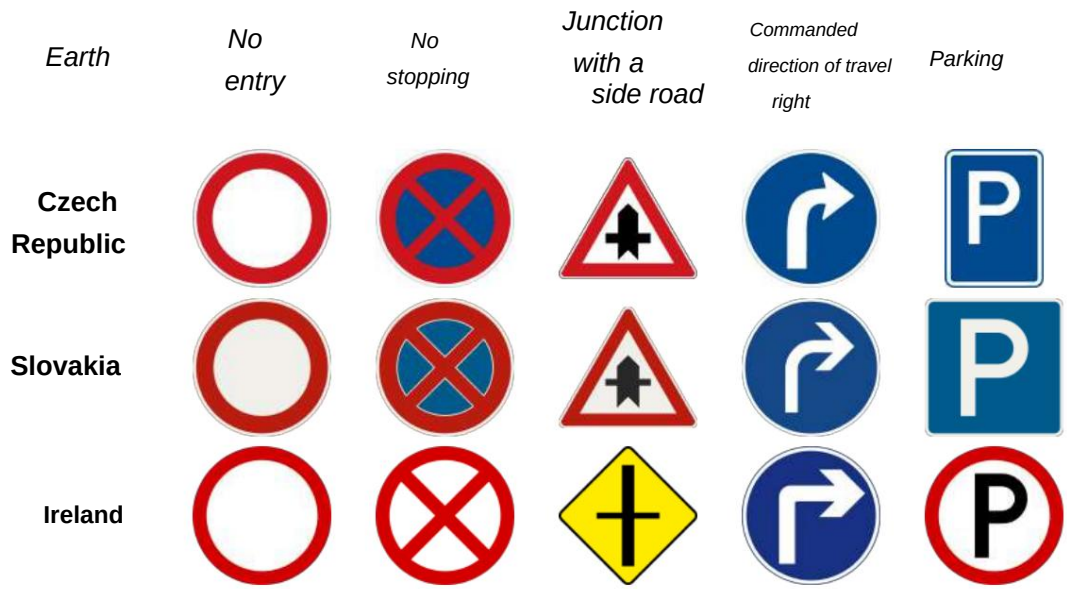


Figure 4.3: Comparison of a sample of road signs of three different countries defined by their laws. Both the Czech Republic and Slovakia are part of *the Vienna Convention on Road Signs and Signals.* Nevertheless, minor differences can be noticed between individual brands. However, the vast majority of brands are very similar. Ireland, which is not part of it, uses road sign standards that are quite different from those used in other countries. *Brand images taken from www .wikipedia .org*

Chapter 5

Architecture of used models and creation of datasets

The entire system is implemented in the **Python31** language . The generative model is implemented using the *TensorFlow2* library in version *2.4.1,* using *CUDA3* technology to accelerate calculations on the graphics card and *cuDNN4 ,* providing efficient implementations of neural network layers. To manipulate the graphic data, I used the *OpenCV* library and for the detector I used the *TensorFlow Object Detection API* 6 framework , which provides the implementation of models for object detection and the scripts needed for their training and evaluation. However, the model_main_tf2.py script had to be modified slightly, because in its original version it incorrectly allocated memory on the graphics card. The system consists of a total of two modules. One day it implements operations with neural networks, and the next it implements operations with slices and the entire data set. The functionality of these modules is then used in the **train.py script,** which allows you to configure and train the generative model and **createTFR.py,** which is able to create training datasets for the detector in TFRecord format. This script is capable of generating TFRecord records from both synthetic and original data and their combinations.

5.1 Generative model

When augmenting the detection dataset, it is important to know the class and location information of the objects in the synthetic images. Control over the class of the generated object can be ensured by using a conditional network, *see section 3.2.* Preservation of the position of the object in the image can be achieved by transforming only the area in which the object is located. Then just insert the transformed area back into the image in the same place. This will keep the object in its position, but change its appearance. However, during this process, it is imperative that the generative model does not change the background, as the data would not look natural after inserting the object back into the original frame. These anomalies would only be in the area of the object, and the detector could learn to locate objects based on their presence, making it useless in real-world examples.

1Available at: python.org/
2Available at: tensorflow.org/
3Available at: developer.nvidia.com/cuda-downloads
4Available at: developer.nvidia.com/cudnn 5Available at: opencv.org/ 6Available at: github.com/tensorflow/models/tree/master/research/object_detection

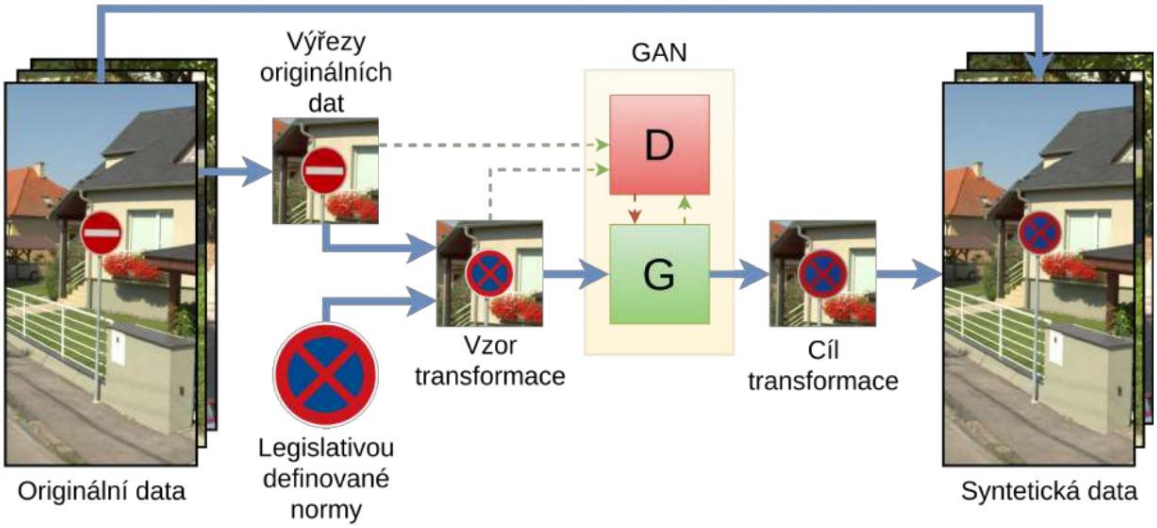


Figure 5.1: A diagram of the generative model is shown in the figure. Solid arrows describe the process of generating synthetic data. First, a road sign is cut from the original image. This is then covered with a standard of another brand of the same shape. The overlaid image is then processed by a generative model that transforms the embedded image into a realistic-looking traffic sign. The output of the generator is inserted back into the scene, giving us a new training example for the detector. Each road sign in the original data set can therefore be transformed into a sign of a different class, but of the same shape. Dashed arrows describe the learning process of the generative model. The discriminator assesses the realism of the original and transformed slices, the PatchGAN architecture, *see section 5.1.2 ,* however, when evaluating the realism of the input, unlike WGAN, also provides the discriminator with a transformation pattern.
Based on the output of the discriminator and the loss function of the generator, the generator weights are adjusted. The discriminator is trained separately.

5.1.1 Creation of a training data set for a generative model

Each traffic sign is made according to a model defined by legislation. In Slovakia, they are defined *by Decree of the Ministry of the Interior No. 9/2009 Coll.* z, the content of which is freely available here7 . This can be used to create a training dataset for a generative model. By cutting out the marks from the original scenes, we get the transformation targets. These targets correspond in appearance to the images that the generator should synthesize. To train a neural network for data transformation, each target example needs to be assigned its pattern. At best, a generative model learns a generalized way of transforming data from the sample domain into data corresponding to the target domain. I created the pattern dataset by overlaying the target data with a standard image of the same class mark, defined by legislation, and transformed to copy the outline of the original mark, *see Figure 5.4.* The norm images are slightly modified before being inserted into the viewport using random noise, contrast change, and random multiplication of the entire image matrix by a number from the interval ÿÿ 0.9, 1.1 ÿÿ . The resulting pair data set is then provided to the gene
rative model during training.

After the model is successfully trained, the set of sample training images can be expanded by overlaying the tag viewport with the image of the tag norm of another class. The sample cutouts are then processed

7 Content available at: https://www.ssc.sk/sk/Technicke-predpisy-rezort/zoznam-vl.ssc

by the generative model and its output is inserted back into the original scene in which the original viewport is located, *see Figure 5.1.* For the purposes of training and validating the system, a non-public data set from Slovak Roads was used.

Perspective transformation

A perspective transformation is a linear projection of a three-dimensional body onto a surface. Due to the three-dimensional nature of the space in which the road signs were captured, this is a suitable method of transforming the road sign norm to overlay the sign in the original cut-out, *see Figure 5.4.* The perspective transformation matrix can be calculated if we know the positions of the four points before and after the transformation. The advantage of the data set from Slovak roads is the annotation of objects using *keypoints, see figure 5.3. Keypoints* define objects of simple shapes with the help of points by which the shape can be precisely determined, including its position in the frame.

For example, an n-gon is annotated with points at its vertices. Most marks are quadrangular or circular in shape, which are annotated with four points. These can be used when calculating the transformation matrix. However, the triangle marks are annotated with only three points, so for the purpose of transformation, I only chose the vertex points at the base of the triangle, and the other two target points are calculated as the midpoints of its arms, *see Figure 5.2.*

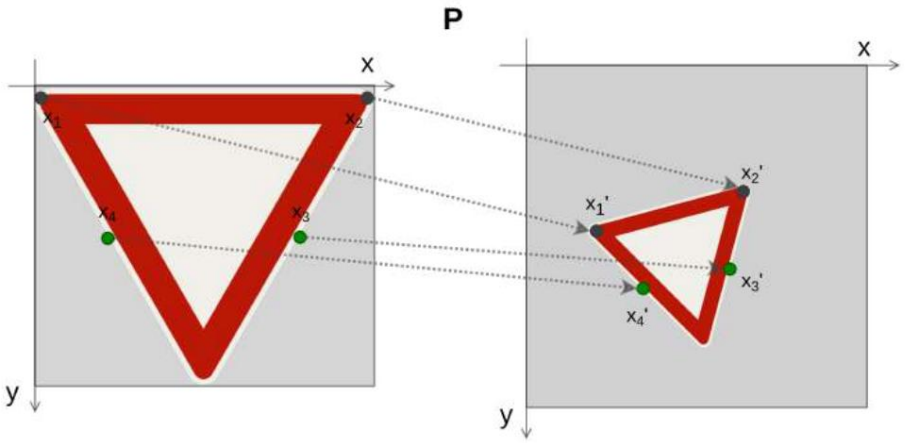


Figure 5.2: Example of using a perspective transformation during pattern creation for a generative model. The mark standard image is first resized to match the mark cut-outs. Then, depending on the shape of the mark, four sample points are determined for the transformation, which are located in the image of the standard at the same place of the mark as the target points in the original cut-out of the mark. From transformation 1, 2, 3, 4, these points, we calculate the transformation matrix and by applying it, we get an image of the mark that exactly overlaps the mark in the cut-out.

Figure 5.3: Example of road sign annotation from the used dataset using two methods.
On the left, the object is annotated with *keypoints,* on the right with *a bounding box.* It can be noted that, unlike the bounding box, annotation using *keypoints* provides information about the exact position of the mark in space.

Figure 5.4: In most cases, using an affine transformation to project a two-dimensional object onto a plane is sufficient. However, in some cases, when the captured mark is close to the camera and its third dimension is noticeable in the image, it does not provide a good enough overlay of the original mark.

Characteristics of the training data set for the generative model

Some viewports where the mark extended beyond the edge were filtered out because they are not suitable examples for a generator that learns to modify only the objects in the middle of viewports and change their surroundings as little as possible. In addition, viewports with markers captured from behind were filtered out, as well as viewports with marker classes that change their appearance within their class, such as city names or the maximum load capacity of a bridge. It is difficult to match these marks with the corresponding image of the standard. In total, the Slovakian Roads dataset for training the generative model contains 1554 pairs of sample and target data at a resolution of 128x128, which provides enough information about the viewport when it is inserted back into the scene so that the resulting image does not look blurry.

5.1.2 Discriminator architecture

The choice of discriminator can have a great influence on the quality of the generated data [2]. In this work, I used *PatchGAN discriminators,* evaluating the realism of images by their parts, *see section 3.2* and critics approximating *the Earth-Mover* distance between synthetic and original data, *see section 5.1.2.*

PatchGAN

The PatchGAN network discriminator, unlike the ordinary discriminator, evaluates the reality of the input image by its parts. At the input, it accepts a sample and an original or synthetic image. In the original publication Pix2Pix [16] , its authors came to the conclusion that on data with a resolution of 256x256, the architecture that evaluates the reality of the image in parts of size 70x70 achieves the most satisfactory results. However, in this work I chose 128x128 resolution data, and therefore it is possible that the model will achieve the best results in a different configuration. The transformation of the data by the generative model is very mild and only applies to certain areas of the tag slice, so I performed a series of experiments with the architectures shown *in Figure 5.5, see experiment 6.4,* to see which configuration would provide the best solution. To avoid tuation, when the discriminator learns to evaluate the reality of the input too well, thereby stopping the training of the generator, its loss function is multiplied by 0.5 when calculating new weights, which slows down its training, but provides a sufficient gradient for training the generator. The average absolute error [16] is used as the loss function of the generator.

When calculating the new generator weights, a combination of the generator loss function and the discriminator output is used in a ratio that can also affect the quality of the generated images.
In the original Pix2Pix publication [16], the authors achieved the best results by using a ratio of 100:1 in favor of the lossy generator function.

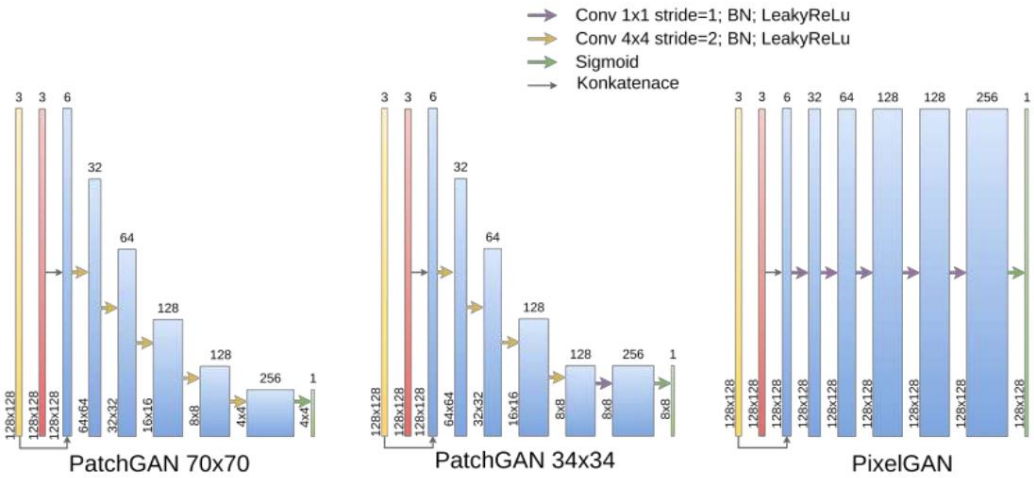


Figure 5.5: Diagram of the architecture of PatchGAN and PixelGAN networks (a special case of Patch GAN, evaluating the reality of the image by individual pixels). The yellow layer corresponds to the input in the form of a sample and the red in the form of an evaluated image. These are combined and several convolutional layers are applied to them. The size of the filter and the step of the convolutional layer determines the size of the evaluated parts of the image. The last convolutional layer is followed by the sigmoid activation function, the output of which is a matrix of values approximating the reality of individual parts of the images.

WGAN

Another GAN architecture used in this work is WGAN, *see 3.1.* Its input is a 128x128 image and its output is a single value approximating the Earth-Mover distance [2]. This network contains only convolutional layers, batch normalization and *LeakyReLu* activation function with parameter = 0.2 [2]. A change in the architecture of the critic or a change in the frequency ratio of the adjustment of the critic and generator weights can affect the quality of the images. The architecture used in this work is shown in Figure 5.6. When implementing the calculation of the loss function and adjusting the weights of the generative model, I was inspired by an existing solution8 .

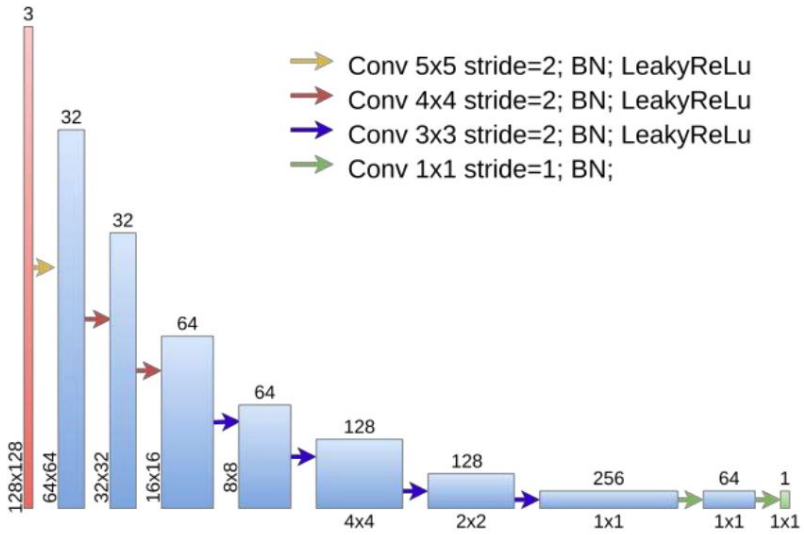


Figure 5.6: Schematic of the architecture of the critic used in the experiments in Chapter 6. The dimensions of the input image are gradually reduced to a feature map of size 1x1x256 using step 2 convolutional layers. Two more convolutional layers with a 1x1 filter and a step of 1 are then applied to this, reducing it to a scalar value approximating the Earth-Mover distance.

8Available at: kaggle.com/amanooo/wgan-gp-keras

5.1.3 Generator architecture

In this work, I used several generator architectures based on the U-Net model [16], *see section 3.2.* The specific models used may differ from each other in their depth (number of encoders and decoders), the use of *dropout* layers and the architecture of functional blocks.

U-Net

U-Net is the simplest of the architectures used. In the first part, using convolutional layers with a larger step, it gradually reduces the resolution of the input image, and in the second part, using convolutional layers with a fractional step, it increases the resolution to the original state, see section *3.2 .*
All ordinary U-Nets used in the experiments in Chapter 6 contain 6 encoders and 6 decoders, *see Figure 5.7.* Unless otherwise stated, a *dropout* layer with parameter = 0.5 is used in all decoders.

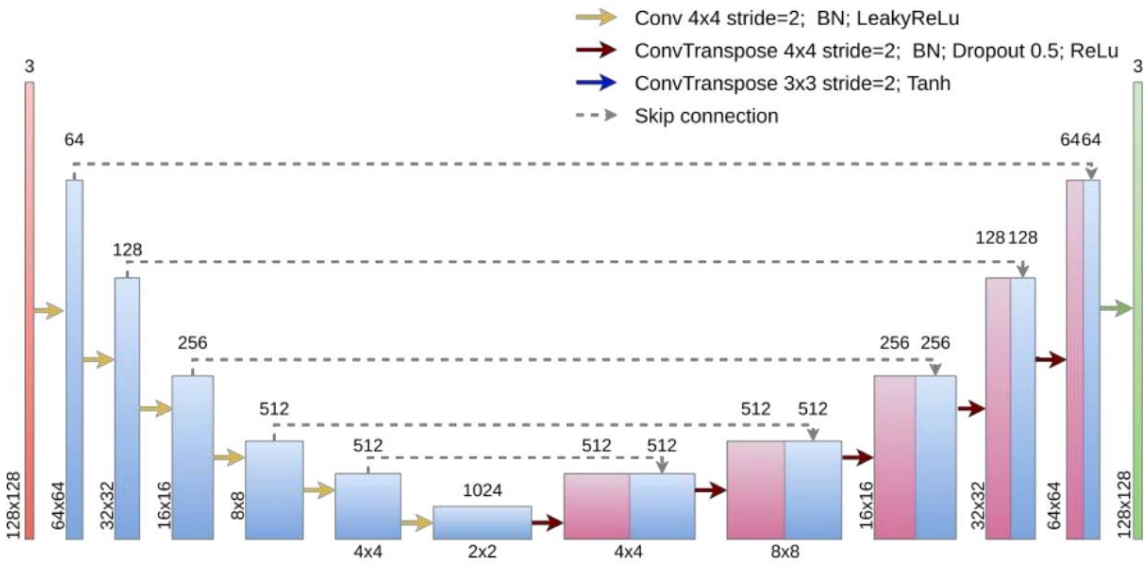


Figure 5.7: Functional diagram of the U-Net architecture used.

RU-Net and Dense RU-Net

RU-Net and Dense RU-Net, *see Section 3.2,* are extensions of the U-Net architecture with residual blocks, *see Figure 5.8.* A total of 3 types of functional blocks were used in the models during the experiments. The first of them is the original residual block [13] and the second is the so-called *pre-activation* residual block, see section 3.2 and figure 3.6, which should achieve better results [14]. Furthermore, DenseNet blocks were used in several variants with a different number of sub blocks, *see Figure 3.7.* In all blocks, the activation layers are followed by a *dropout* layer with a parameter of 0.8.

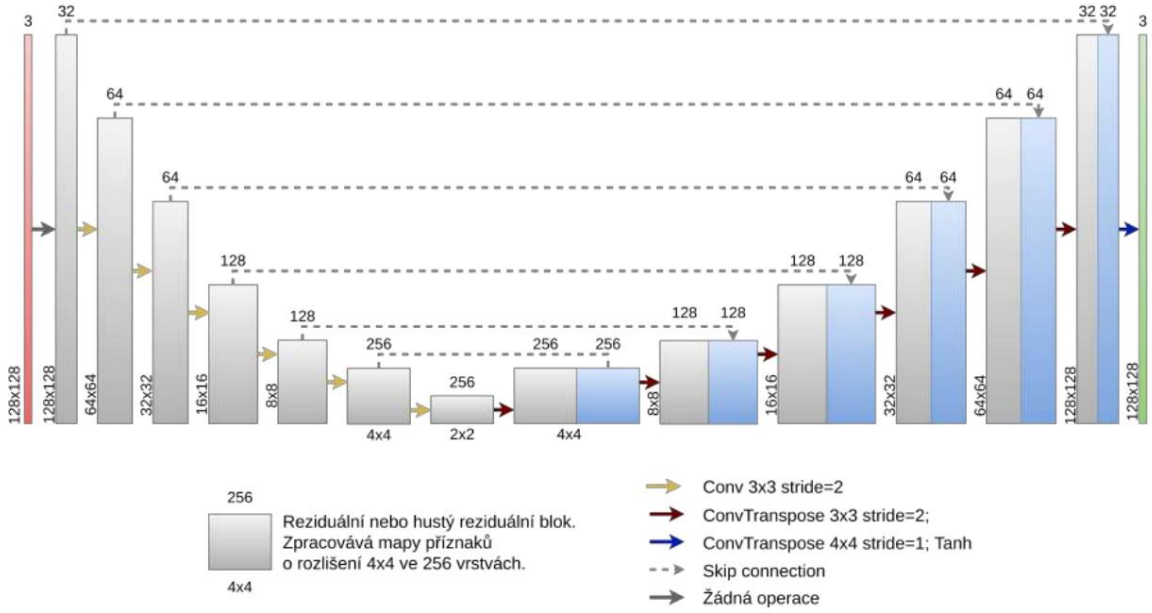


Figure 5.8: Generalized RU-Net and Dense RU-Net network diagram. The network, like U-Net, uses *skip connections* and also gradually reduces the input image using convolutional filters with a step of 2 to a very small feature map up to the so-called bottleneck and then, again using convolutional filters with a fractional step, transforms it into an image the same dimensions as the input. However, in contrast to the U-Net architecture, each convolutional layer is preceded by a residual or DenseNet block. The models used differ mainly in the architecture of functional blocks and their number.

5.2 Detector model

As a model for traffic sign detection, I chose the neural network of the SSD architecture, see section 2.1, namely its *open source* implementation *SSD MobileNet V2 FPNLite 640x640,* which is publicly available z9 . The application of this model to the problem of traffic sign detection is suitable, mainly due to its high speed and low memory requirement. The SSD architecture manages to evaluate images several times per second and is capable of real-time object detection [22], which is a key feature of a traffic sign detector.

5.2.1 Detector configuration

The model was configured to apply augmentation during the learning process in the form of random adjustment of brightness, hue and tint and with a 50% probability of vertical rotation of the image. Random horizontal rotation of the image for some classes of marks will cause a change in its semantic meaning, unlike the vertical one, so I did not include horizontal rotation in the image augmentation options. Furthermore, the aspect ratios of the default SSD bounding boxes were adjusted to 0.5, 0.75, 1, *see section 2.1,* according to the information shown in graph 5.9. At is

9Available on address: https://github.com/tensorflow/models/tree/master/research/object_detection

their setting must take into account the data transformation from *1000x500* to *640x640* due to the fixed configuration of the detector input.

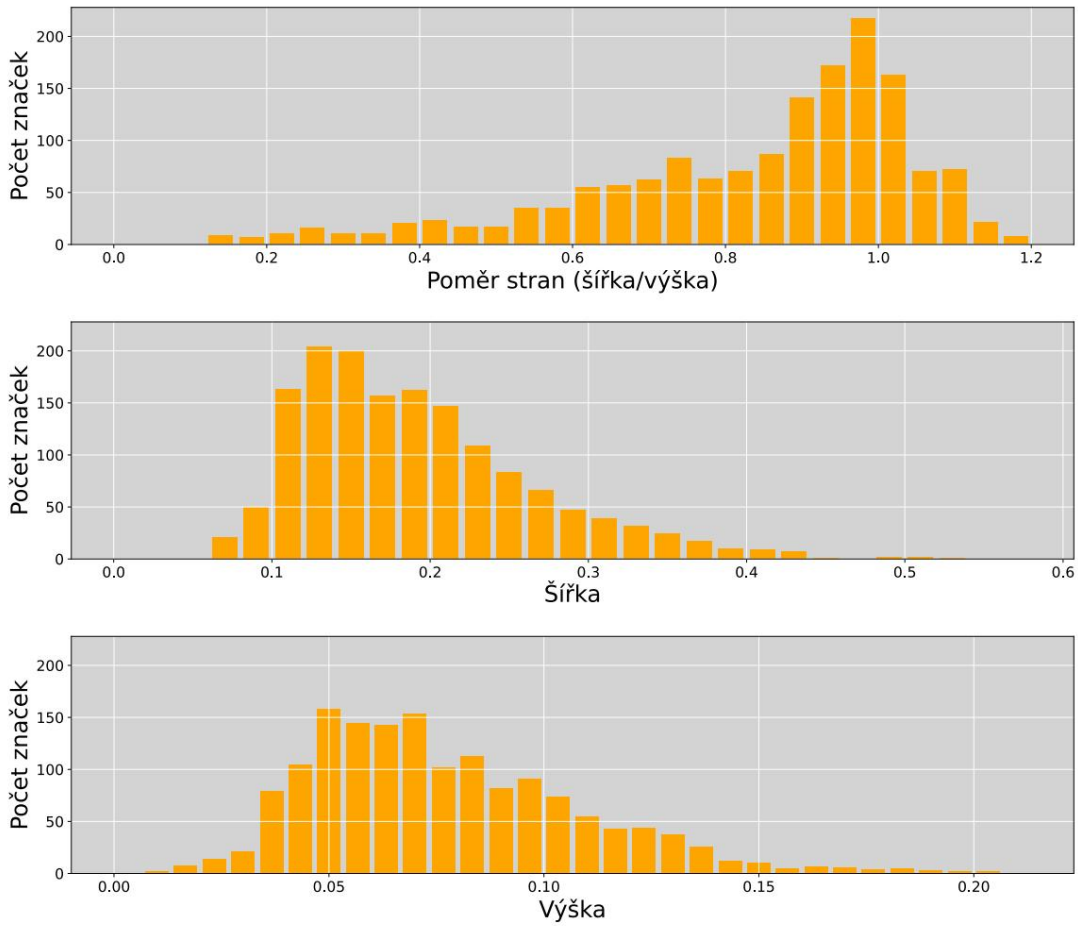


Figure 5.9: Graph showing information about the Slovak Roads data set used. According to this information, the configuration of the generation of the default bounding boxes for the SSD model can be estimated and tuned [22]. On the first graph, you can see the distribution of the ratio of the width of the objects to their height. The next 2 graphs show the relative distribution of individual dimensions (height and width). size to the total size of the images in a given dimension.

5.2.2 Creating a training dataset for the detector

The dataset used in this work contains 6225 images and 246 different marker classes. Subsequently, it was reduced only to traffic signs, the content of which does not change across its instances, because the generative model works only with such signs, of which the detector is a metric.
The dataset modified in this way contains 1554 images with a total of 145 road sign classes and was used for training the reference model for detecting road signs. The SSD model achieved a reference accuracy of **50.2%** mAP for >= 0.5 on the unexpanded data set, *see graphs 6.8.* This data set was , subsequently expanded with the same number of synthetic examples for each brand class using generative models. Synthetic and original datasets and their combinations were experimented with in Section 6.2.

Chapter 6

Experimentation

In this chapter, I deal with the configuration and evaluation of experiments with a generative model and a detector. I verify the assumptions from the previous chapter 5 and try to find the most suitable configuration of both models.

6.1 Experiments with the generative model

In order to create a generative model capable of synthesizing visual quality data, several experiments were performed with different discriminator and generator architectures and their different configurations. The generative model was trained after 64 batches, with the Adam optimizer with parameters 1 = 0.5 and 2 = 0.9. The learning rate is set to 0.002 for the [illegible] and 0.001 for the discriminator. The weights of all layers are initialized through a normal distribution with a center at point 0 and a deviation of 0.02. During learning, pairwise data are augmented by contrast variation, affine transformation, and random multiplication and addition to individual pixel values. Within a data pair (pattern and target), the augmentations are always exactly the same to keep the data pairs consistent.

Background adjustment

Absolute avoidance of background modification by combining the original image and generated data from only the annotated tag area is not very effective [37]. Due to imprecise annotations, it is impossible to completely cover the mark with the image of the standard, *see Figure 5.4,* where parts of the original mark can be seen even when using the perspective transformation. Therefore, it is necessary for the generator to modify the immediate surroundings of the mark, while leaving the more distant surroundings untouched. The results of the experiments showed that the generative model is capable of such a transformation if it is provided with suitable pairs of data containing consistent examples of the given transformation during training, *see Figure 6.1.*

Data batch size during training

The authors of the *Pix2Pix* architecture recommend using batches of images of size 1 during training [16]. However, the use of such a dose leads to unnecessary overhead of the program and the training process takes many times longer, compared to the use of a larger dose. Therefore, I performed a series of experiments, *see Figure 6.1,* to verify if it is possible to obtain a quality generator output using a larger dose. All generative models in this chapter were trained for 500 epochs, so the number of steps depends on the size of the dose used, *see table 6.2.* In these experiments

a PatchGAN 70x70 discriminator was used, *see 5.1.2* and a U-Net generator containing *dropout* layers with a parameter of 0.5, *see section 5.1.3.* Examples of synthetic data from experiments are in Table 6.1.

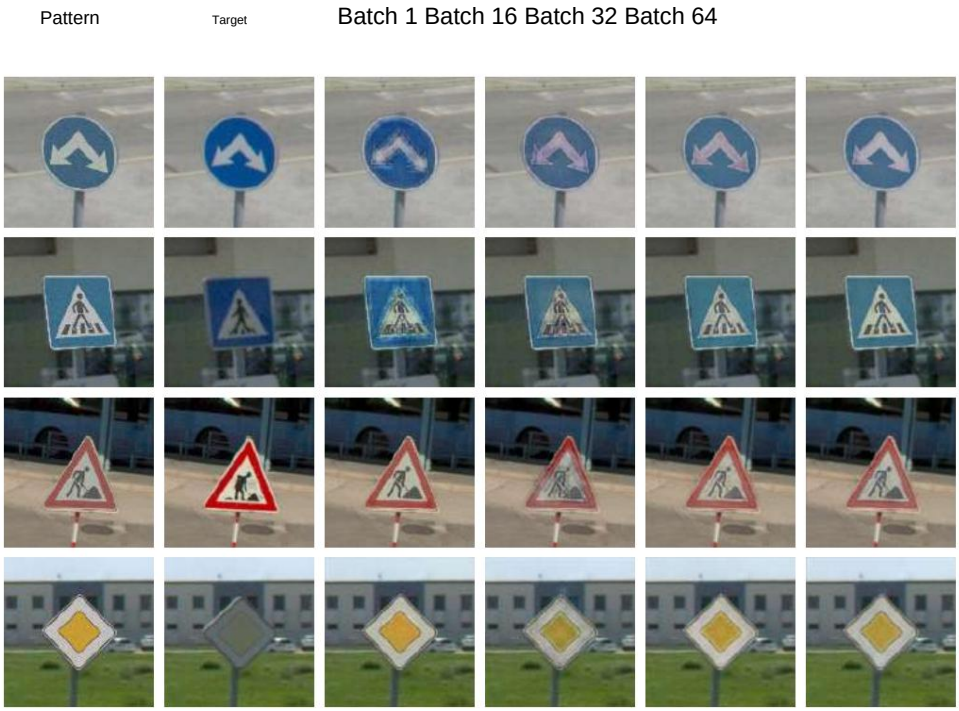


Figure 6.1: Example of transformation of a sample image by tested generators. The results of the experiment show that using a batch size of one generates some marks that are significantly deformed. Using a higher dose slightly improves the visual quality of the generated data, but the outputs are more pattern-like than targets. The experiment showed that it is possible to train the generative model after larger batches, without noticeable losses in the quality of its output.

Dose size Training time Number of steps		
	13:49:38	770000
1	1:58:58	48500
16	1:37:12	24000
32 64	1:29:51	12000

Figure 6.2: Comparison of training times and number of steps needed to complete 500 epochs using different batch sizes.

Based on the results of this experiment, I am in all subsequent experiments used a 64 size dose during training.

Variation of generated data

During the training of GANs, several failure states can occur, *see section 3.1,* when the generative model produces, for example, data only from a limited probability distribution of the training data. However, for the purpose of synthesizing a quality dataset, the generative model must be able to generate even different-looking instances of objects of the same classes. A certain variation in the output data can be ensured by the *dropout layer,* which has the same input and output connections, and its parameter determines the probability that the connection will pass through the *dropout* layer to the next layer. Another source of variability in the generated data is the random adjustment of the mark standard before it is inserted into the cutout, *see section 5.1.1,* which creates variation not only at the output of the generator but also at its input. Experiments focused on the *dropout layer effect,* as well as all subsequent experiments, include these modifications to the standards. They differ only in the parameter value of this layer. An experiment with data created without modifying the norm images uses dropout layers with a parameter of 0.5. Examples of synthetic data from experiments are in Table 6.3.

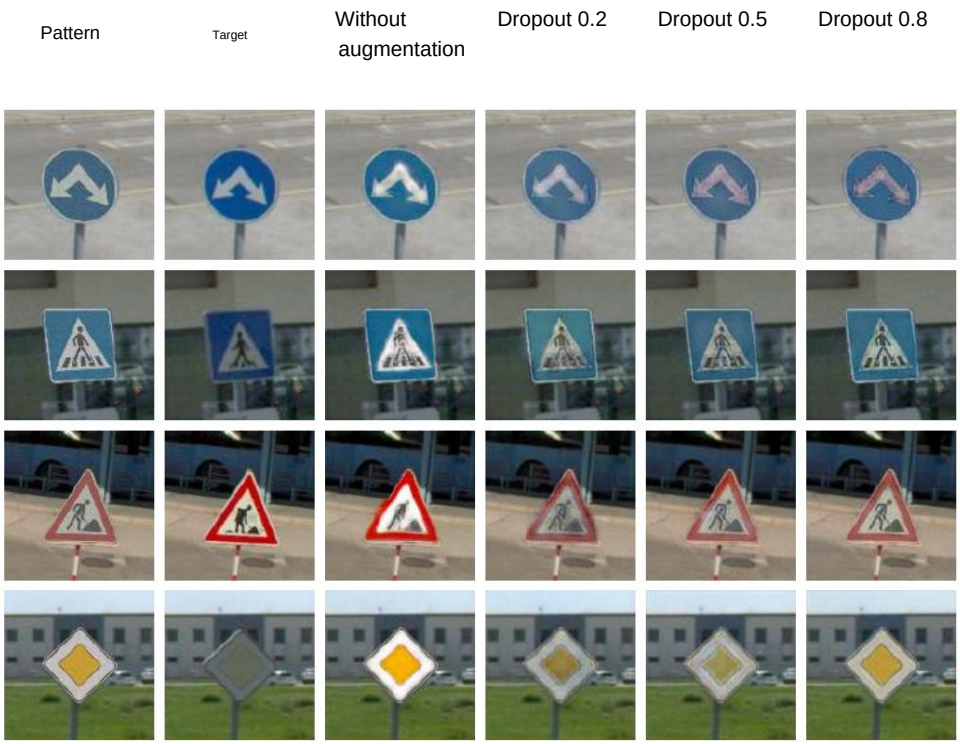


Figure 6.3: Sample transformation of a sample image by tested generators. The experiment showed that the use of *dropout* layers in the U-Net decoder with a suitable parameter improves the quality of the results. Visually, the best results were achieved using a *dropout* layer with a parameter of 0.8. A generator trained on a data set created without modification of standard marker images generates blurred and otherwise distorted markers, implying an improvement in output quality by applying these modifications.

Based on the results of this experiment, I am in all subsequent experiments used *dropout* layers with a parameter of 0.8.

Experiments with different discriminator architectures

The quality of the generated data can be improved by choosing a different discriminator [16, 2]. The *PatchGAN* type discriminator evaluates the realness of the image with a matrix of values representing the realness of slices of dimensions *NxN* [16], *see section 3.2,* and its special variant *PixelGAN* evaluates slices of size *1x1.* The size of the dimension can affect the quality of the generated data, so I performed a series of experiments with discriminators evaluating slices of size *70x70, 34x34, 1x1 (pixelGAN), see section 5.1.2.* The other two experiments were performed using WGAN, *see Section 3.1* and were focused on the effect of the ratio of the critic and generator weight adjustments on the quality of the generated data. Examples of synthetic data from experiments are in Table 6.4.

Figure 6.4: Example of transformation of a sample image by tested generators. For the *Patch GAN* network, the quality of the generated data is fairly consistent for all configurations used, but the *34x34* configuration slightly outperforms the other configurations in terms of output quality. Further experiments showed that the ratio of the adjustments of the critic and generator weights has an effect on the quality of the generated data. In the ratio 3 configuration, in favor of the critic, there is a distinct periodic noise at the output of the generator, but in the ratio 5 configuration the noise is negligible. Using WGAN can achieve more realistic-looking outputs, but the PatchGAN method has less effect on changing the background of the image.

Based on this experiment, I chose a *34x34* configuration for PatchGAN and a 5:1 critic-generator weight adjustment configuration for WGAN in all subsequent experiments.

Ratio of loss function weights during generative model training

Another parameter of the PatchGAN network that can affect the quality of the generated data is the ratio of the loss functions of the generator and the discriminator when adjusting the generator weights, *see section* 3.2. The original Pix2Pix architecture weights a loss function of a generator 100 and a discriminator 1 [16] in solving tasks from semantic scene segmentation to colorization of black and white images. However, the transformation of road signs is of a much milder nature, and therefore it is likely that the model will achieve the highest quality outputs with a different configuration.
Examples of synthetic data from experiments are in Table 6.5.

Figure 6.5: Sample transformation of the sample image by the tested generators. The results of the experiments showed that this parameter has a relatively large influence on the quality of the generator output. The generator achieves visually the highest quality outputs in the 150:1 loss function ratio configuration, and qualitatively very similar outputs are also achieved in the 100:1 configuration. As the weight of the generator loss function decreases, the model becomes more focused on fooling the discriminator at the expense of the quality of the generated data.

Based on this experiment, I chose the PatchGAN discriminator configuration in all subsequent experiments, which adjusts the generator weights from the loss functions of the generator and the discriminator in a ratio of 150:1.

Experimenting with residual models and PatchGAN

Another way to improve the quality of synthetic data is to use a combination of residual or DenseNet blocks with U-Net, *see Section 5.1.3.* The first four experiments were performed with the RU-Net network, of which the first two networks contained the original residual blocks and the other two *pre-activation* residual blocks (marked PA in the table). Experiments with the same architecture of residual blocks differ from each other in the depth of the network used, where the first set of experiments uses RU-Net with 8 residual blocks and the second with 12 residual blocks. Deeper networks should achieve better results [13] and the *pre-activation* residual block architecture should generate better data compared to the original architecture [14]. In the next two experiments, I used two Dense RU-Net networks with 12 DenseNet blocks, which differ in the number of sub-blocks.

The aim of these experiments is to determine the effect of the number of sub blocks on the quality of the network output.

All the aforementioned generative models were trained using PatchGAN, *Figure 6.6* and WGAN, *Figure 6.7,* configured according to the results of previous experiments.

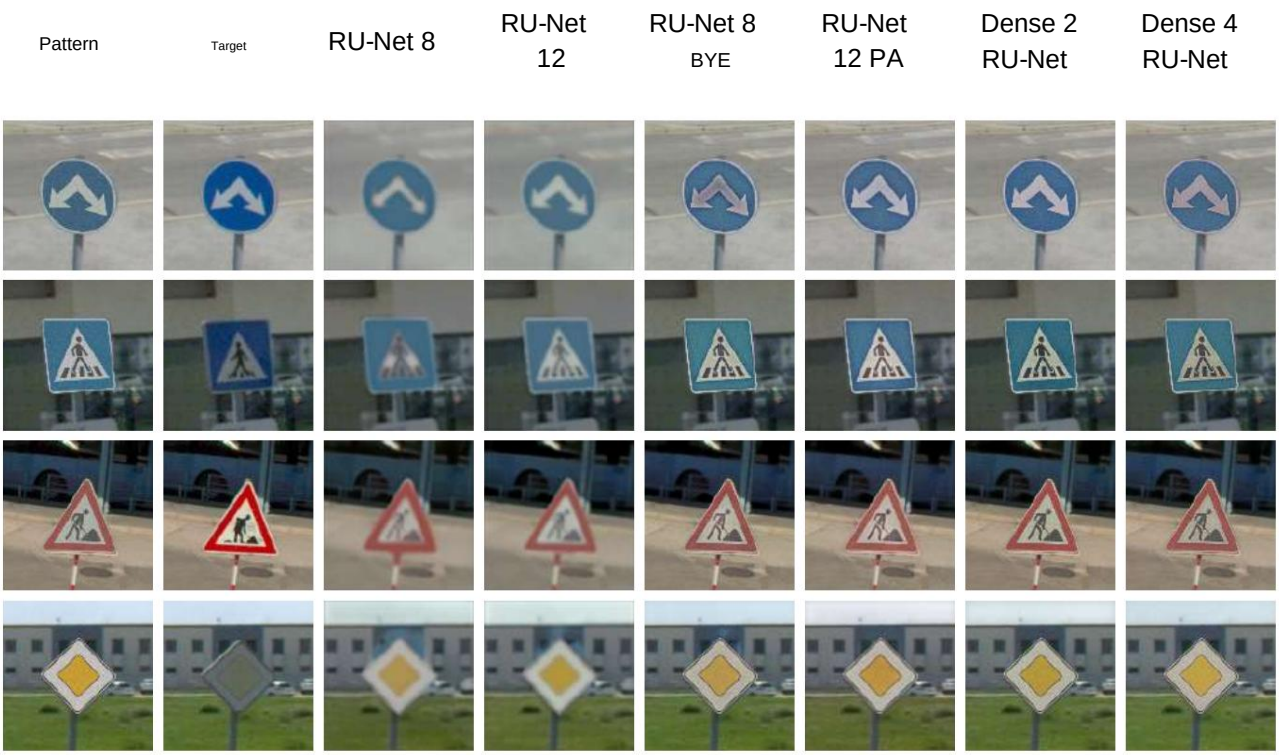


Figure 6.6: Example of transformation of a sample image by tested generators. Experimental results show that deeper RU-Net networks achieve better results. Although the classic RU Net generates very blurred and low-quality data, the use of the *pre-activation* residual block greatly improves the quality of the synthetic data. However, Dense RU-Net networks achieve the highest quality output. By comparing the outputs of both Dense RU-Net networks, we find that the number of sub blocks in this case has no visible effect on the resulting image quality.

Experimenting with residual models and WGAN

I performed the last set of experiments with the same generator configuration as in the previous experiment. The only difference is that they are trained using the WGAN method. Critic scales were adjusted 5x more often than generator scales.

Figure 6.7: Example of transformation of a sample image by tested generators. As with the use of PatchGAN, it is evident from the experimental results that the depth of the residual network has a slight effect on the quality of the generated data. However, in this case, the use of the *pre-activation* block leads to a slight deterioration of the results. Although the generated data looks distorted and of lower quality than when using PatchGAN, the resulting images are much more similar to the target ones. However, Dense RU-Net can generate relatively realistic images compared to other generator architectures in this experiment. By using 4 sub blocks instead of 2, there is no visible change in data quality.

Although none of the above models provide a perfect transformation from a standard to a realistic-looking road sign, for the purpose of training the detector, it is possible to synthesize the data using several generative models and then combine them. Although the data synthesized by the WGAN-trained generator looks more realistic, it contains a relatively large amount of unwanted background transformations. On the other hand, the output of generators trained with PatchGAN keeps the background relatively intact, but the resulting label is more similar to its pattern than to the target.

It is by combining these two models and their characteristics that we can achieve mutual complementation of these key features in a synthetic data set and thus train a more accurate model for object detection.

6.2 Experiments with the mark detector

In all experiments with the detector, it was trained for 25000 steps, in batches of 8 frames.
The detector recognizes a total of 145 classes from 640x640 images . The first 1000 steps are trained with a low learning rate of 0.025, which is then increased to 0.08 and gradually decreases to zero at step 25000. The slope of the optimizer is set to 0.9. The first experiment was aimed at determining the reference accuracy of a detector trained on original data. The reference data set is described in more detail in section 5.2. It consists of 1554 images, of which 300 are used for model validation. The metric used in quantifying the accuracy of the detector is mAP@.50IoU - mean Average Precision counting only detections for which >= 0.5 applies (hereinafter referred to as mAP).

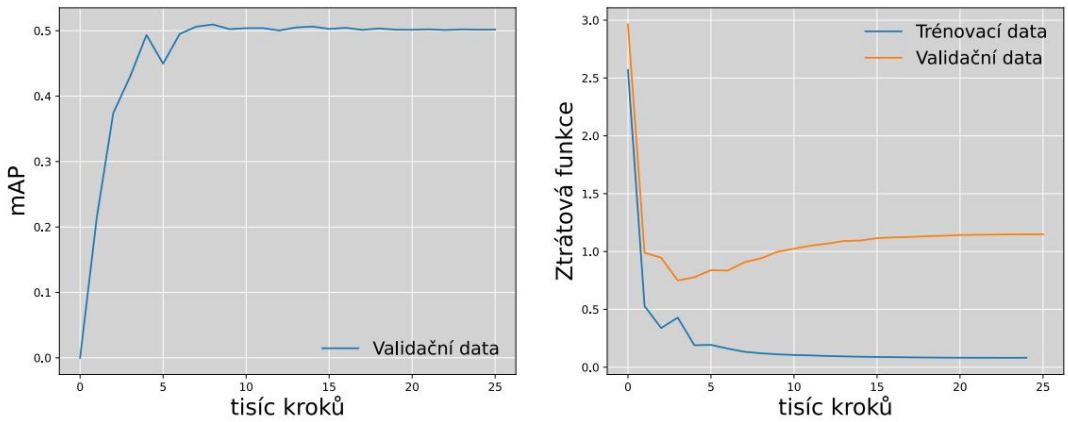


Figure 6.8: The first plot shows the evolution of *mAP* on the validation data and the second plot shows the combined detection and classification loss function during training on the validation and training data. A slight overtraining of the network is evident on the loss function difference.

Experimenting with the generator with augmented datasets

The accuracy of the detector achieved in the previous experiment is relatively low because the dataset contains a relatively large number of classes and a small amount of data. Some classes are contained very sparsely in the dataset. The system proposed in section 5 is best designed for those cases where a certain class occurs rarely or not at all in the data set, because it synthesizes objects based on their defined norm, thus eliminating the need for a sufficient number of examples of objects of each class, for its quality synthesis . To expand the original dataset, I chose two generative models that had the best visual output in the experiments in Chapter 6. Both use the DenseNet architecture, described in Section 3.2, with two sub-blocks. The network contains a total of 12 DenseNet blocks. All ReLu activation layers in the generator are followed by a *dropout* layer with parameter 0.8, *see 6.3.* The first model was trained using the PatchGAN discriminator in a 34x34 configuration, *see Figure 6.6* with a generator loss function weight of 150, *see Figure 6.5.* The second model, identical in generator architecture, was trained through a critic approximating the Earth-Mover distance. The critic to generator training ratio was set to 5. A synthetic was created using each of the models

dataset, including 20 instances of each class tag, i.e. a total of 2900 annotated objects. Information about these datasets is in Table 6.1.

Origin	Number of examples
Original	1554
WGAN	2900
PatchGAN	2900

Table 6.1: All data sets used can be divided into these 3 basic ones. During the experiments, the detectors were trained on these data sets and their combinations.

Experimenting with synthetic datasets

As part of this experiment, 4 detectors were trained on extended datasets, *see table 6.2.* The aim of the experiment is to verify whether the accuracy of the traffic sign detector can be improved by using a generative model. The course of the detector training process is shown in graphs 6.9 and 6.10.

Origin	Number of mAP@.50IoU examples	
Original	1554	50.2%
Original+WGAN	4454	76.4%
Original+PatchGAN	4454	75.2% **80.1%**
Original+PatchGAN+WGAN	7354	

Table 6.2: Experimental results show that using the generative model leads to more accurate detections. They further show that the best results can be achieved by combining multiple generative models.

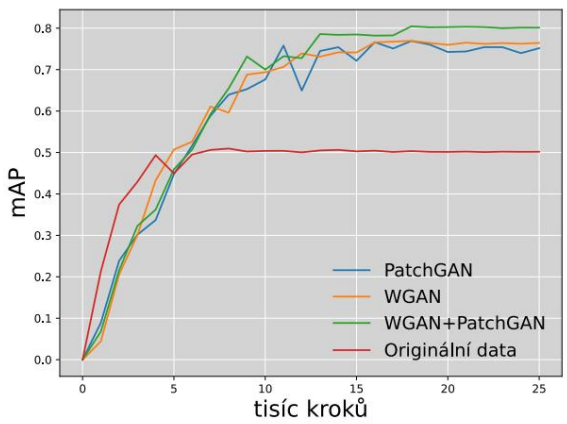


Figure 6.9: The graph shows the evolution of *mAP* during detector training on all data sets from Table 6.2. A drastic increase in the accuracy of the detector can be observed for all extended data sets, by combining both generative models, only a relatively small improvement can be achieved compared to the use of a single one.

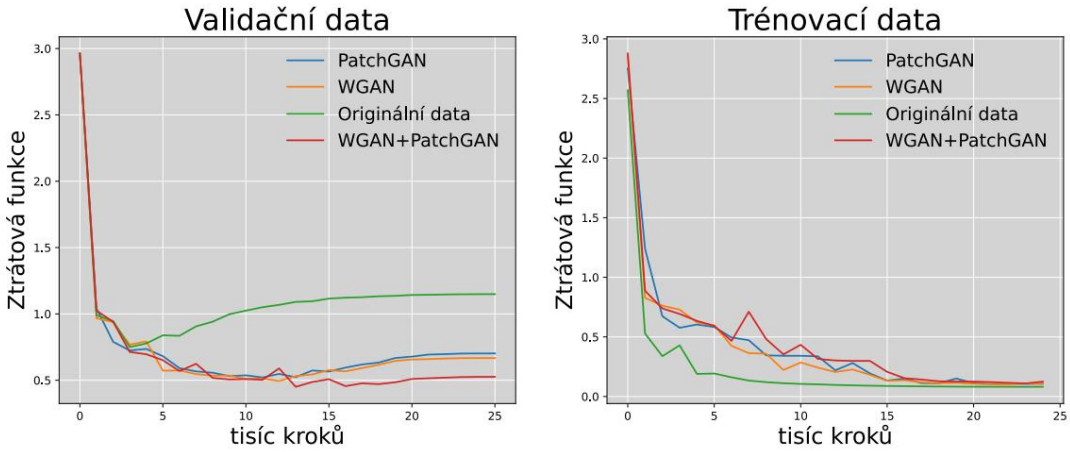


Figure 6.10: The graph shows the evolution of the loss function during detector training on all data sets from Table 6.2. It can be noticed that the training process on all extended datasets converges more slowly, but achieves a lower overtraining rate.
This behavior is typical during model training on datasets of different sizes, confirming the quality of the synthetic data.

Experimenting with fully synthetic datasets

In chapter 4.2 I describe the problems with the detection of road signs across different countries, the biggest of which are the different standards defining their appearance. The system implemented in this work is designed to be able to generate realistic-looking instances of traffic signs from a norm image inserted into the scene. This can be used precisely to synthesize a road sign detection dataset for any country.
The aim of this experiment is to determine the accuracy achieved by a detector trained only on synthetic data, which can be used to verify the system's ability to generate high-quality data sets, for example for the detection of traffic signs defined by different standards. The progress of the detector training process is shown in graphs 6.11 and 6.12.

Origin	Number of examples	mAP@.50IoU
Original	1554 4454 4454 7350	50.2%
WGAN		30.7%
PatchGAN		48.3%
PatchGAN+WGAN		**59.6%**

Table 6.3: The results of the experiment show that even the detector trained only on synthetic data achieves a relatively high accuracy. Although using one of the generative models is not enough to match the accuracy of the detector trained on the original data, combining the two models can achieve a 9.4% larger mAP.

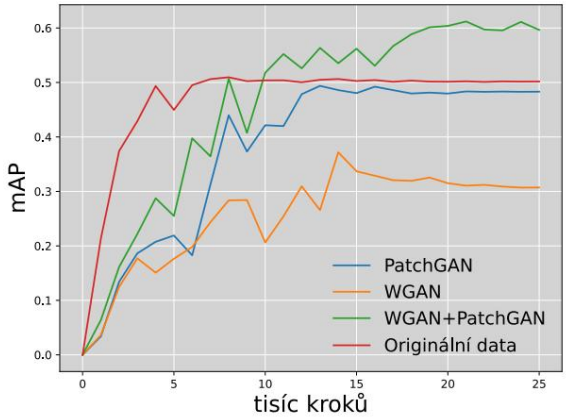


Figure 6.11: The graph shows the evolution of *mAP* during detector training on all data sets from Table 6.3

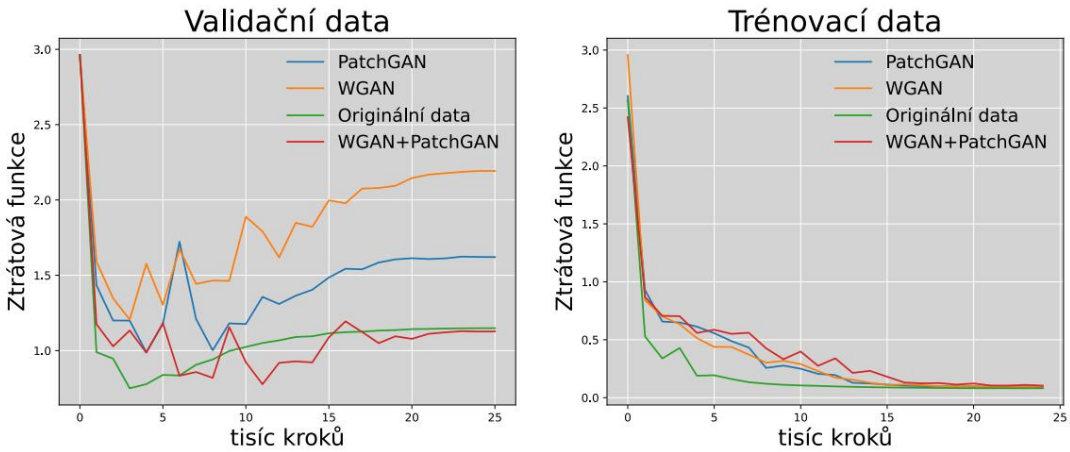


Figure 6.12: The graphs show the evolution of the loss functions of validation and training data during detector training on all data sets from Table 6.3

A detector trained on original data converges much faster than detectors trained on synthetic data. This may be due to the uneven selection of examples of different classes during training on the original data, which makes the model learn better and faster to recognize frequently occurring classes of marks. By combining both generative models, the generator achieved the best result, but even using only the PatchGAN generator, the discriminator achieved almost the same accuracy as when trained on the original data. A detector trained only on the WGAN-generated dataset achieved very poor results, probably due to a large modification of the background of the markers, which was not compensated for by selecting other examples during training. This is also evidenced by the difference between the loss function on the validation and training data, which in this case is probably not caused by overtraining the model, but by a larger difference between these data.

6.3 Other experiments

The system described in section 5.1.3 transforms images of objects inserted into real scene viewports into real-looking instances of this object in the original scene at the same location. The training data set from Slovak roads for the generative model, *see section 5.1.1,* was created with the aim of transforming the image inserted into the scene into a realistic-looking object, while keeping its content well distinguishable. This leads to the possibility of using the functionality of the model even for classes that were not part of the training process. This principle can be used to expand the data set not only by additional instances of class objects that are in the original data set, but also objects that were not captured in it. Another possible use is the creation of a data set, for example, of another country that uses different traffic sign patterns, as experiment has confirmed that it is possible to train a relatively accurate detector even on a completely synthetic data set, see Figure *6.11* .

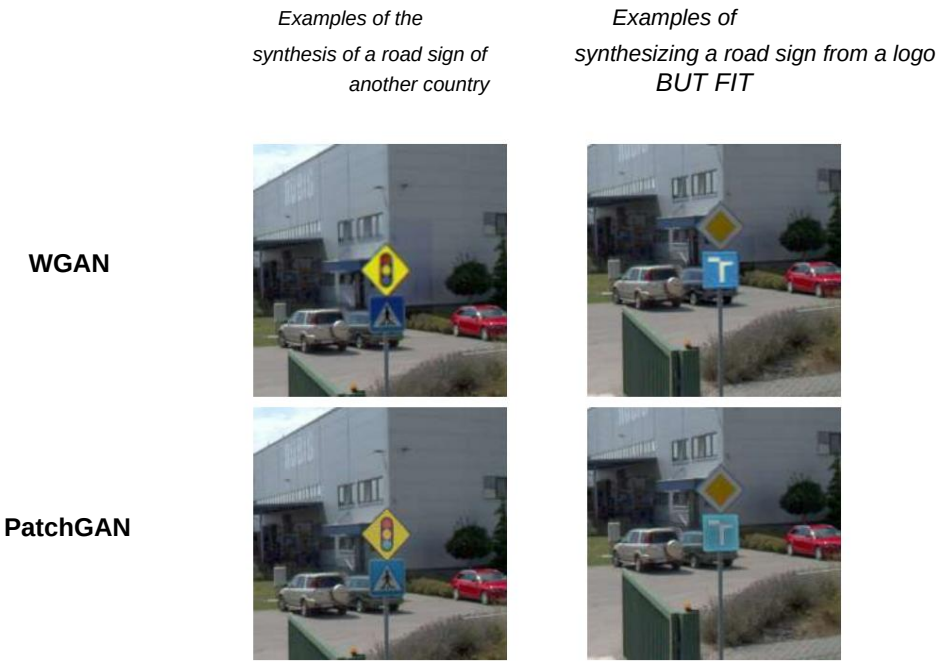


Figure 6.13: An example of the extended use of the proposed model. The input of the generative model is an image, so we have quite a lot of control over the generated data and its content. This can be used to create a synthetic data set containing road signs of any appearance.

Chapter 7

Conclusion

The aim of this work was to design a system based on generative neural networks GAN, capable of expanding the existing data set for traffic sign detection with high-quality synthetic images containing objects and rarely occurring classes. At the beginning of the work, modern object recognition methods based on deep convolutional neural networks, such as SSD or Faster R-CNN, are described, as well as methods for quantifying their accuracy.

Next, I describe generative methods based on the competition of two neural networks (GANs). Here I describe several GAN architectures, for example DCGAN or Conditional GAN, I describe different procedures for their training and the recommended configuration for their convergence. I also focus on the possibilities of image transformation using the Pix2Pix model and combinations of residual neural networks with the U-Net network. Furthermore, I deal with difficulties in the detection of road signs and the possibilities of how to eliminate these difficulties as best as possible by using generative models.

Furthermore, all the architectures of neural networks used in the generative model and the detector are described and their advantages and disadvantages are mentioned. Here I also propose several options for their configuration in order to improve the quality of synthetic images and make detection more accurate, which are checked in the next chapter and are either incorporated into the model or discarded. The characteristics of the training data sets for both models and the process of their creation are also described.

During experimentation with the generative model, its outputs are continuously evaluated and tuned, finally two models generating the best quality images (Patch GAN and WGAN) are selected, which are used for the synthesis of the artificial data set. Subsequently, several detectors were trained on original, synthetic, and combined datasets, and subsequently their accuracy was evaluated by the mAP metric (.50 IoU), in order to validate the generative model. Using only synthetic data to train the detector model, the mAP reached 59.6%, which is 9.4% more than the detector trained on the original data. When using a combination of original and synthetic data, the detector reaches 80.1% mAP, which is 29.9% more.

Further development could focus on the design of a single generative model that will combine the strengths of PatchGAN, i.e. preserving the background of the object during transformation, and WGAN, i.e. generating very realistic-looking objects. Such a model would further improve the accuracy of the detectors and would be able to create very realistic datasets. It would also be very interesting to extend the system with the possibility of generating a sign standard by inserting text and pictograms, which would make it possible to synthesize even more complex, specific traffic signs.

Literature

[1] Arcos García Álvaro, García, JA Álvarez and Soria Morillo, LM Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing.* 2018, Vol. 316, pp. 332–344. DOI: https://doi.org/10.1016/j.neucom.2018.08.009. ISSN 0925-2312. Available from: https://www.sciencedirect.com/science/article/pii/S092523121830924X.

[2] Arjovsky, M., Chintala, S. and Bottou, L. *Wasserstein GAN.* 2017.

[3] Bansal, M., Krizhevsky, A. and Ogale, A. *ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst.* 2018.

[4] Dai, J., Li, Y., He, K. and Sun, J. *R-FCN: Object Detection via Region-based Fully Convolutional Networks.* 2016.

[5] Dalal, N. and Triggs, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).* 2005, Vol. 1, pp. 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[6] de la Escalera, A., Moreno, LE, Salichs, MA and Armingol, JM Road traffic sign detection and classification. *IEEE Transactions on Industrial Electronics.* 1997, Vol. 44, No. 6, pp. 848–859. DOI: 10.1109/41.649946.

[7] Efros, AA and Freeman, WT Image Quilting for Texture Synthesis and Transfer. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques.* New York, NY, USA: Association for Computing Machinery, 2001, pp. 341–346. SIGGRAPH '01. DOI: 10.1145/383259.383296. ISBN 158113374X. Available from: https://doi.org/10.1145/383259.383296.

[8] Everingham, M., Van Gool, L., Williams, CKI, Winn, J. and Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision.* June 2010, Vol. 88, No. 2, pp. 303–338. DOI: 10.1007/s11263-009-0275-4. ISSN 1573-1405. Available from: https://doi.org/10.1007/s11263-009-0275-4.

[9] Girshick, R. *Fast R-CNN.* 2015.

[10] Girshick, R., Donahue, J., Darrell, T. and Malik, J. *Rich feature hierarchies for accurate object detection and semantic segmentation.* 2014.

[11] Goodfellow, I. *NIPS 2016 Tutorial: Generative Adversarial Networks.* 2017.

[12] Goodfellow, IJ, Pouget Abadie, J., Mirza, M., Xu, B., Warde Farley, D. et al. *Generative Adversarial Networks.* 2014.

[13] He, K., Zhang, X., Ren, S. and Sun, J. *Deep Residual Learning for Image Recognition.* 2015.

[14] He, K., Zhang, X., Ren, S. and Sun, J. *Identity Mappings in Deep Residual Networks.* 2016.

[15] Huang, G., Liu, Z., Maaten, L. van der and Weinberger, KQ *Densely Connected Convolutional Networks.* 2018.

[16] Isola, P., Zhu, J.-Y., Zhou, T. and Efros, AA *Image-to-Image Translation with Conditional Adversarial Networks.* 2018.

[17] Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. *University of Toronto.* May 2012.

[18] Krizhevsky, A., Sutskever, I. and Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems.* January 2012, Vol. 25. DOI: 10.1145/3065386.

[19] Lee, H., Kim, J., Kim, EK, and Kim, S. Wasserstein Generative Adversarial Networks Based Data Augmentation for Radar Data Analysis. *Applied Sciences.* 2020, Vol. 10, No. 4. DOI: 10.3390/app10041449. ISSN 2076-3417. Available from: https://www.mdpi.com/2076-3417/10/4/1449.

[20] Lienhart, R. and Maydt, J. An extended set of Haar-like features for rapid object detection. In: *Proceedings. International Conference on Image Processing.* 2002, Vol. 1, pp. I–I. DOI: 10.1109/ICIP.2002.1038171.

[21] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R. et al. *Microsoft COCO: Common Objects in Context.* 2015.

[22] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. et al. SSD: Single Shot MultiBox Detector. In: Leibe, B., Matas, J., Sebe, N. and Welling, M., eds. *Computer Vision - ECCV 2016.* Cham: Springer International Publishing, 2016, pp. 21–37. ISBN 978-3-319-46448-0.

[23] Lowe, D. Object Recognition from Local Scale-Invariant Features. *Proceedings of the IEEE International Conference on Computer Vision.* January 2001, Vol. 2.

[24] Mirza, M. and Osindero, S. *Conditional Generative Adversarial Nets.* 2014.

[25] Padilla, R., Netto, SL and da Silva, EAB A Survey on Performance Metrics for Object-Detection Algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP).* 2020, pp. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.

[26] Padilla, R., Passos, WL, Dias, TLB, Netto, SL and Silva, EAB da. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics.* 2021, Vol. 10, No. 3. DOI: 10.3390/electronics10030279. ISSN 2079-9292. Available from: https://www.mdpi.com/2079-9292/10/3/279.

[27] Radford, A., Metz, L. and Chintala, S. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.* 2016.

[28] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. *You Only Look Once: Unified, Real-Time Object Detection.* 2016.

[29] Ren, S., He, K., Girshick, R. and Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2017, Vol. 39, No. 6, pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[30] Ronneberger, O., Fischer, P. and Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* 2015.

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

[32] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S. et al. *ImageNet Large Scale Visual Recognition Challenge.* 2015.

[33] Sanchez, J. and Perronnin, F. High-dimensional signature compression for large-scale image classification. In: *CVPR 2011.* 2011, pp. 1665–1672. DOI: 10.1109/CVPR.2011.5995504.

[34] Simon. *Opel Insignia will feature 'Opel Eye' camera.* TopSpeed, 2008. Available from: https://www.topspeed.com/cars/car-news/opel-insignia-will-feature-opel-eye camera-ar59270.html.

[35] Srivastava, A., Valkov, L., Russell, C., Gutmann, MU and Sutton, C. *VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning.* 2017.

[36] Uijlings, J., Sande, K. van de, Gevers, T. and Smeulders, A. Selective Search for Object Recognition. *International Journal of Computer Vision.* 2013. DOI: 10.1007/s11263-013-0620-5. Available from: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf.

[37] Ševýík, P. *Preparation of training data using generative neural networks.* Brno, CZ, 2020. Bachelor thesis. Brno University of Technology, Faculty of Information Technologies. Available from: https://www.fit.vut.cz/study/thesis/20903/.

[38] Wakabayashi, D. Self-Driving Uber Car Kills Pedestrians in Arizona, Where Robots Roam. *The New York Times.* New York: [bn]. 2018. ISSN 0362-4331.

[39] Wang, S.-F., Yu, W.-K. and Li, Y.-X. Multi-Wavelet Residual Dense Convolutional Neural Network for Image Denoising. *IEEE Access.* 2020, Vol. 8, pp. 214413–214424. DOI: 10.1109/ACCESS.2020.3040542.

[40] Yang, Z., Chai, Y., Anguelov, D., Zhou, Y., Sun, P. et al. *SurfelGAN: Synthesizing Realistic Sensor Data for Autonomous Driving.* 2020.

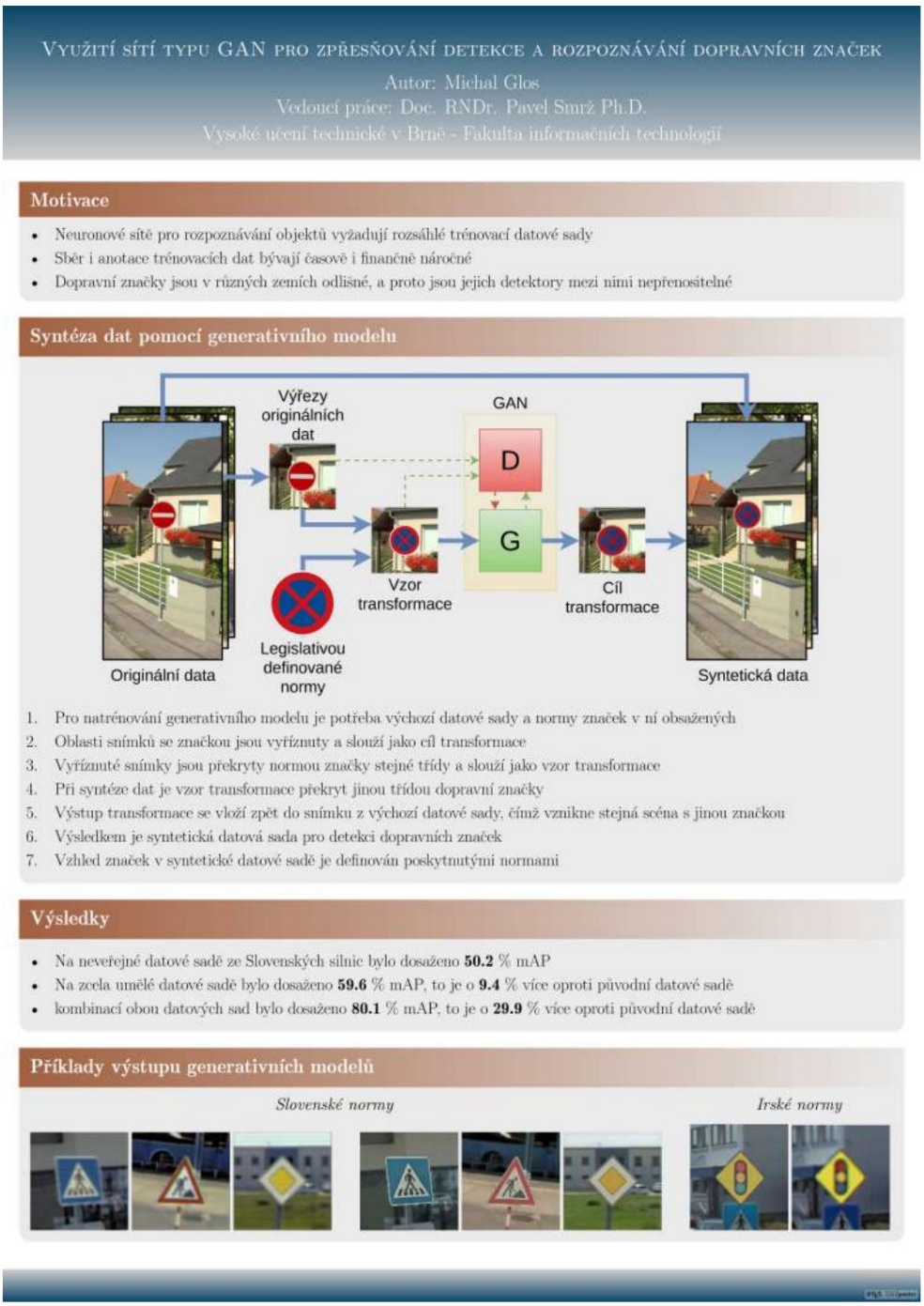
[41] Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T. et al. *LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop.* 2016.

[42] Zhang, Z., Liu, Q. and Wang, Y. Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters.* Institute of Electrical and Electronics Engineers (IEEE). May 2018, vol. 15, No. 5, pp. 749–753. DOI: 10.1109/lgrs.2018.2802944. ISSN 1558-0571. Available from: http://dx.doi.org/10.1109/LGRS.2018.2802944.

List of attachments

Appendix A

Poster



Appendix B

Contents of the attached storage media

- /*Detektor* - Files related to the road sign detection model
- /*GAN* - Files related to the generative model
- /*Latex* - Files with pdf of work and poster including source codes
- /*venv* - Files with Python3 virtual environment configurations
- /*README.md* - File containing installation and startup instructions

More detailed information on the directory structure is contained in the file *README.md*