# Road images augmentation with synthetic traffic signs using neural networks

*A.S. Konushin [1,2], B.V. Faizov [1], V.I. Shakhuro [1]*
*[1] Lomonosov Moscow State University, Moscow, Russia;*
*[2] NRU Higher School of Economics, Moscow, Russia*

***Abstract***

Traffic sign recognition is a well-researched problem in computer vision. However, the state of the art methods works only for frequent sign classes, which are well represented in training datasets. We consider the task of rare traffic sign detection and classification. We aim to solve that problem by using synthetic training data. Such training data is obtained by embedding synthetic images of signs in the real photos. We propose three methods for making synthetic signs consistent with a scene in appearance. These methods are based on modern generative adversarial network (GAN) architectures. Our proposed methods allow realistic embedding of rare traffic sign classes that are absent in the training set. We adapt a variational autoencoder for sampling plausible locations of new traffic signs in images. We demonstrate that using a mixture of our synthetic data with real data improves the accuracy of both classifier and detector.

*Keywords*: traffic sign classification, synthetic training samples, neural networks, image recognition, image transforms, neural network compositions.

*Citation*: Konushin AS, Faizov BV, Shakhuro VI. Road images augmentation with synthetic traffic signs using neural networks. Computer Optics 2021; 45(5): 736-748. DOI: 10.18287/2412-6179-CO-859.

## *Introduction*

Modern computer vision methods are based on machine learning techniques and require labelled datasets for training. The accuracy of the trained model depends on the size and quality of the available dataset. Dataset labelling is a labor-consuming and time-consuming process that is prone to errors. In contrast, synthetic data generation can produce virtually unlimited training datasets without annotation errors. This is why methods for generating synthetic images are actively investigated in recent years.

In this paper, we consider the task of generating artificial data for training traffic sign recognition models. Traffic sign recognition is a significant problem, which gains the stable interest of researchers over the years. Traffic sign detection and classification are used in driver assistance systems, self-driving cars, for maintaining up-to-date high-resolution maps and traffic sign inventory. Modern open datasets for traffic sign recognition can contain thousands of frames with two hundred classes. However, a distinctive feature of the traffic sign recognition problem is a significant amount of rare classes. Objects of such classes can either be present in small amounts in datasets or absent. But it is still required to train recognition algorithms for such traffic sign classes, since the importance of rare classes on the road is no less than that of frequent.

We investigate modern methods for generating synthetic training data using neural networks. Since even state-of-the-art methods are unable to generate the whole photos of the traffic scene with photo-realistic quality, we propose to embed artificial signs in real images. Two questions arise immediately: how to make the inserted object consistent in appearance with the scene and where to position it.

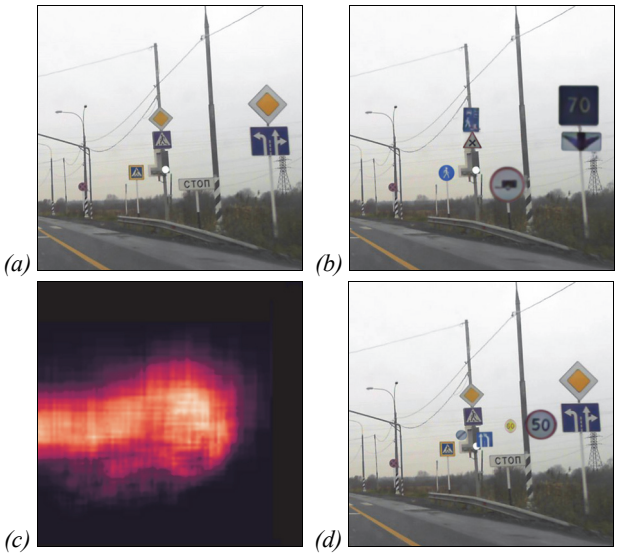


*Fig. 1. Example of fragment with 6 traffic signs. Here on one fragment real ones are replaced with new synthetic. On another fragment there are embedded new signs: (a) original image; (b) real signs replaced with synthetic; (c) predicted sign location heatmap; (d) additional synthetic signs*

We focus on the recognition of rare traffic sign classes. Since such signs are absent or limited in the real dataset, we can't directly train a neural network to generate images of such signs. Instead, we aim to create a synthetic traffic sign processing method that will improve the realism of simple synthetic images obtained from the sign icon. We propose three processing methods based on generative adversarial networks [1, 2, 3]. Since modern neural network detectors such as Faster R-CNN analyze the whole image at once, this images should look realistic. We embed artificial signs in real images instead of already existing traffic signs. To do this, we first remove

the existing ones via inpainting and then place synthetic signs on their places (see fig. 1*b*). Inpainting is done using a neural network that is trained separately or jointly with the sign processing method. Such a technique allows us to augment images with rare sign classes with the correct geometric placement and evaluate the individual contribution of object processing methods.

In the second part of our work, we adopt a method based on variational autoencoder [4] to predict the correct location and size for synthetic traffic sign insertion. To predict plausible traffic sign placement in a frame, we first automatically obtain semantic segmentation of the image and then sample locations using variation autoencoder. An example of the obtained heatmap is shown in fig. 1*c*. After obtaining locations, we insert synthetic traffic signs in addition to real traffic signs in a frame (see fig. 1*d*).

Overall, we propose three methods for processing synthetic traffic signs and a new method for their placement on real road images at the geometrically correct position. Proposed methods allow augmenting the real road images with high-quality synthetic traffic signs for classes, which are absent in the real training dataset. We have conducted an extensive experimental evaluation of the proposed methods. It has demonstrated that usage of generated data improves the quality of traffic sign detection and classification, especially for the rare classes.

## *1. Related work*

### *1.1. Synthetic image generation and processing*

The augmentation of real images with new synthetic objects can be implemented using different methods. The simplest and most obvious way is to draw an object without any processing [5, 6]. However, this approach will lead to unrealistic images and does not allow to obtain high-quality synthetic samples. Recently generative adversarial neural networks [1] have been applied to such problems. Such methods perform image processing so that artificial objects matches the background in colour and lighting [7, 8, 9]. However, the geometric position and shape of the embedded objects are still not taken into account.

The basic idea of a generative adversarial network is to have two separate parts – a generator and a discriminator. The generator creates synthetic images. Discriminator learns to distinguish generated images from real ones. These neural network's components try to deceive each other during the training process. In [10] for image generation, convolutional neural network architecture with transposed convolutions was proposed to increase the resolution of generated images. The proposed approach with convolutional layers made it possible to train the neural network faster and improve the quality. Other authors [11] used the Laplace pyramid and several generators and discriminators. Also, the researchers proposed work on the conditional generation of an object with a given class [12]. The generator receives not only random noise at the input but also the class label of the object to generate.

GAN models have been successfully applied for image transfer between domains. One of the notable examples is CycleGAN [2], which doesn't need labelled pairs of images from the source and target domains for training. It has two generators and two discriminators. Suppose we have two image domains $\mathcal{A}$ and $\mathcal{B}$. The first generator learns to transfer images from $\mathcal{A}$ to $\mathcal{B}$, and the second generator on the contrary from $\mathcal{B}$ to $\mathcal{A}$. First discriminator trains to distinguish synthetic and real images from $\mathcal{B}$, and the second discriminator vice versa. During the inference process, only the desired generator is used.

The rapid progress of GANs is quite astonishing. A StyleGAN architecture was proposed [3], which demonstrates a surprisingly realistic generation of people's faces. They were generated from random vectors, which were at first transformed by a small fully-connected part of a neural network to obtain a vector in intermediate latent space. The Adaptive Instance normalization (AdaIN) layers [13] are used in the generator to transfer information from vector in latent space. Also, random noise is actively added to the architecture in the intermediate layers to obtain a variety in the small details of the individual generated images. Our proposed methods for high-quality synthetic traffic sign generation are based on this approach.

The previous methods don't predict the location of embedded objects. In the paper [14], the authors suggested the adversarial approach for generating synthetic object placement and processing. A proposed neural network has a branch for predicting the location and size of a new object. A simple colour correction of six predicted parameters is used for the first stage of object processing. Then a refinement network is used to improve object consistency with the background. As usual, architecture has discriminator for distinguishing synthetic image and new segmentation network, which learns to predict a mask of the artificial object.

The usage of synthetic training data for accuracy improvement of recognition models is actively investigated. In [15] the quality of the re-identification of people in video was improved by adding synthetic data to real data. In [16] synthetic data were added to the training set to improve the quality of liver lesions classification. In papers [17, 18], game engines are used to generate the labelled city scenes. Synthetic data made it possible to improve the quality of the final algorithm and reduce the requirements for the amount of real data by three times.

The authors of paper [19] suggested generating synthetic road images from the GTA computer game by transferring data from one domain to another. As a target domain, they used images from the Cityscapes [20] dataset. Their approach is based on CycleGAN architecture.

### *1.2. Predicting synthetic object locations*

Most modern neural network architectures for changing position and parameters of objects are based on Spatial Transformer Networks [21]. The idea of such architectures is to add a separate part of a neural network that

will generate affine transformation parameters for an added object according to the background. The resulting affine transformation is applied to a grid of pixels that specify where and which pixel of an object will be positioned. The article shows that these transformations are differentiable and can be optimized in neural networks.

Spatial Transformer Networks became popular for synthetic object placement. Authors of works [14, 22] based their approaches for generating locations of new entities by predicting affine transformations. Besides discriminator during the training process, these methods relied on an additional signal such as image segmentation network or target classifier / detector networks. Disadvantages of a spatial transformer are bad convergence, instability, and complex training process.

Article [4] proposed a VAE-based approach for object placement on road images. The algorithm has two separate modules for determining *where* and *what* could be placed in the picture. A generator with Spatial Transformer Networks is used in the first module to determine where to place the object. In the second module, there is a generator for the shape of an embedded object to determine what exactly needs to be placed.

### *1.3. Traffic sign generation and recognition*

The traffic sign recognition methods have a long history. Early approaches were based on finding corners and feature points in images [23]. Usage of synthetic training datasets has been investigated since 2007 [24]. Generation of syn thetic examples for training traffic sign classifiers has been implemented by applying affine transformations to sign pictograms.

In [25] a four-stage system for detection and classification of traffic signs was proposed. It included a cascade detector and a set of neural network classifiers for each type of traffic sign. This model was trained with synthetic data proposed in paper [26]. A suggested approach for the generation of synthetic data used heuristic methods, based on computer graphics. But it also tried to predict the best parameters for current data. Paper showed that models trained on synthetically generated data could produce good results.

Since the introduction of modern deep learning methods, they have been applied to the traffic sign recognition. In the paper [27], the authors first collected a massive training dataset and then proposed a fully convolutional simple neural network architecture for the simultaneous detection and classification of traffic signs.

In [28] synthetic traffic signs with poles were generated by computer graphics methods and then improved with a neural network, base on CycleGAN. To preserve the traffic sign class while processing an additional identity loss has been using during the training of this model. Then artificial traffic signs were embedded into reals photos. Simple heuristics based on a reconstruction of camera parameters and simple 3D-modelling was used to determine new signs location. Experimental evaluation showed that this approach works better compared to random object placement. However, the best results were obtained if new artificial traffic signs replace existing ones.

Currently, the best results in traffic sign recognition are achieved by adapting modern detection architectures. For example, in [29, 30] anchor-based methods have been used, with specific optimizations for speed. In [29] authors used the ResNet-50 as the backbone to build a pyramidal feature network. In [30] MobileNet-backbone with suggested Localization network is used.

Conventional convolutional neural networks can be used for traffic signs classification, such as AlexNet[31], ResNet[32] etc. In 2019 the article [33] proposed to apply the special pre-processing procedure to the traffic signs before classification. Then, processed signs are fed to the input of a small neural network, based on LeNet.

Different convolutional architectures were tested on the traffic sign classification problem in the article [34]. Authors showed that CNN architectures are well suited for this task and drew attention to the lack of real data. The authors concluded that techniques like image pre-processing and data-augmentation are useful to improve classification accuracy.

Another approach aimed at solving the problem of rare traffic signs classification was proposed in [35]. Authors use WideResNet [36], trained with contrastive loss, for feature extraction. One discriminator is used to distinguish rare and frequent classes. Authors proposed to classify frequent classes using features from the last layer of the neural network, and rare classes using the nearest neighbour method.

## *2. Proposed methods*

We explored two different ways to embed traffic signs in images:

- **Replacement of existing real traffic sings with artificial ones**. In this case, we use inpainting at the place of the real sign to generate plausible background. Then artificial traffic sign is embedded on top of it. This way of generating synthetic data allows increase training set with new examples of rare classes with the correct geometric position. The article [28] showed that this approach improves the quality of neural networks for classification and detection. It also allows us to evaluate better the individual contribution of proposed processing methods for improving target neural networks quality. For inpainting we used Edgeconnect [37] architecture.
- **Embedding additional artificial signs in new positions**. In this case, we need to learn how to find the most suitable position for the new traffic signs first, and then perform their processing. To find the correct position of new traffic signs, a neural network architecture based on [4] was chosen.

### *2.1. Processing of embedded traffic signs*

In both ways, we need to process artificial signs to improve visual consistency with the background. We

propose three models for this task. The first two of them are trained together with the inpainting network, and the third is trained separately. The first two models are based on the ideas of CycleGAN, where network performs transferring from the domain of artificial signs to the domain of real ones. The third model is fundamentally different and inspired by the ideas of StyleGAN, in which the neural network itself learns how to generate correct traffic sign icons that are consistent with the background.

All proposed approaches take into account the context of the image around the embedded sign. That is the main difference from existing methods for artificial traffic sign processing.

#### *2.1.1. First approach ("pasted")*

In this approach, we train together neural networks for inpainting and processing of embedded traffic sign. We use Edgeconnect [37] as the basis for the inpainting architecture. We remove the part of the model for object boundaries generation. The whole proposed architecture consists of two generators and two discriminators.

The first generator receives an input image patch of size 128×128 pixels and a mask of a removed part in the middle of it. The output from the first generator inpaints the removed part. The first discriminator receives either the inpainted patch of an image or the original patch without the removed part and learns to distinguish the real ones from the generated ones. During training, this patch is cut out from random places of source pictures, and a random rectangle is removed from it exactly in the middle so that each side of the removed part is not more than 64 pixels.

The icon of a traffic sign is then embedded in the middle of the output of the first generator, so that icon's maximum side is 64 pixels minus a small random number. This patch with an icon is fed to the input of the second generator, which should improve the visual quality of the fragment. The second discriminator receives at the input either output of the second generator, or a real patch with a traffic sign and learns to distinguish them from each other.

Both generators will inevitably change background with uncut part of the image, so the pixels around the cut-out patch are restored by the mask of the removed part.

We chose cross-entropy as an adversarial loss function of both discriminators. Additionally, the first generator has an *L1-loss*, *perception loss* and *style loss* [38] between the inpainted by the first generator patch and the correct image. For the second generator, there is an *L1-loss* for the background around the sign so that it does not change. Also, a *perception loss* was added between the input and output of the second generator and *style loss* between the output of the second generator and the output of the first generator, before embedding the sign icon.

The architecture diagrams of neural networks are shown in figure 2. During inference and generating of the synthetic data set, patches with real traffic signs are cut out of the image and replaced with patches with embedded artificial signs.

#### *2.1.2. Second approach ("cycled")*

In the first approach, real traffic signs in patches are used only in the second discriminator. That means that the first generator performs in painting of patches for which the correct background is known in advance, but the second generator embeds the sign when the true result of embedding is unknown. We decided to add a second data stream to the training process, where the input will be fed with the patch, in which the real sign was previously located, but was cut out. Next, for the second data stream, the inpainting of the cut-out part of the fragment of the picture is performed by the first generator. Here, unlike the first stream, the true output of the first generator is unknown. Then the icon of the sign of the same class, which was in a real patch, is embedded. This icon is processed by the second generator. As a result, the entire neural network should ideally get a picture identical to the original one.

In addition, *L1-loss*, *perception loss* and *style loss* between the outputs of the second generator and the real image were added as loss functions for the second data stream. Also, we added *L1-loss* between the input and output of the first generator around the area of cut out a rectangle in a fragment.

The architecture of the neural network itself does not differ from the first approach. Cross-entropy is used similarly to the first as an adversarial loss function of discriminators in the second data stream.

The scheme of the second data stream is shown in the fig. 3.

#### *2.1.3. Third approach ("styled")*

Two previous models have shown good quality already, but we have decided to use a more advanced generator to push the quality further. Both previous models combine two neural networks for inpainting and processing images, which are trained simultaneously. In this method, we train two parts separately. As a neural network for inpainting, we use an architecture similar to the previous approaches, based on the EdgeConnect.

Let us consider in more detail the second neural network for the processing of embedded signs in patches. We use StyleGAN [3] as the basis for this model. It will not process an icon, which already embedded in the background, but it will generate a traffic sign consistent with the background. To achieve this result, we have made several significant changes to the StyleGAN:

- Instead of generating a feature vector from random noise as in the original fully connected neural network, we propose to use two convolutional neural subnets. The first convolutional subnet gets as input an image of a $64 \times 64$-sized icon embedded in a $128 \times 128$-sized background patch, where the real traffic sign used to be located before. Subnet converts it

to a vector of length 548. The second convolutional subnet receives resized to 64 × 64 input fragment (the real size is 128 × 128) of the background without a sign. At the output from it, a vector of length 64 is obtained. Next, the two resulting vectors are concatenated into the vector v_desc of length 612.

- A simple two-layer classifier has been added, which, using the vector v_desc, tries to determine the class of a traffic sign from 205 possible. This classifier improved the quality of the generated images. It seems to us that this is happening because it regularizes the neural network so that it encodes exactly the properties related to the class of the sign and not its appearance.
- The process of generating a sign does not begin with a trained constant activation 4 × 4 map, but with a map obtained from v_desc using one fully-connected layer.
- An additional second discriminator is added, which distinguishes the synthetic sign embedded into the background patch from the real one.
- As in the original StyleGAN, all parts of the neural network are first trained for small 8 × 8 pictures, then the layers of generators and discriminators are gradually turned on up to the size of the 64 × 64 sign icon located in the center of the background with a size of 128 × 128.

An adversarial loss function was WGAN-GP in both discriminators. Also with VGG13 neural network we added *perception loss* between the output of the neural network and the icon, embedded into the background without processing. Additionally, we used a small weight *perception loss* between the background itself and the output of a neural network with a sign. It has been observed that this increases the realism of the generated images.

During training, synthetic traffic signs are located in places where there used to be real traffic signs. That is why we previously performed inpainting of real signs.

The processing scheme of traffic signs in the third model is shown in fig. 4. The proposed method allowed a generation of images with good quality and exceeded two previous models in experimental evaluations.

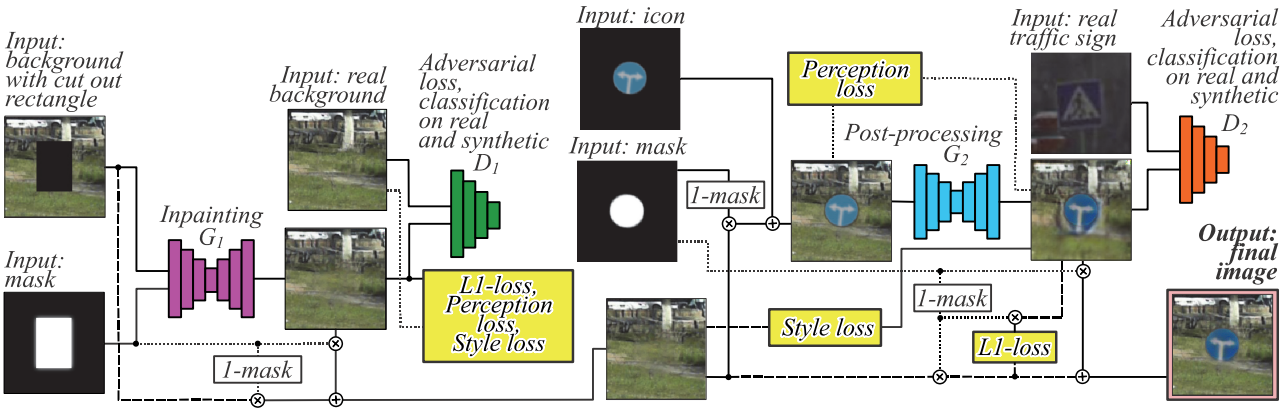


*Fig. 2. The architecture of the first approach*

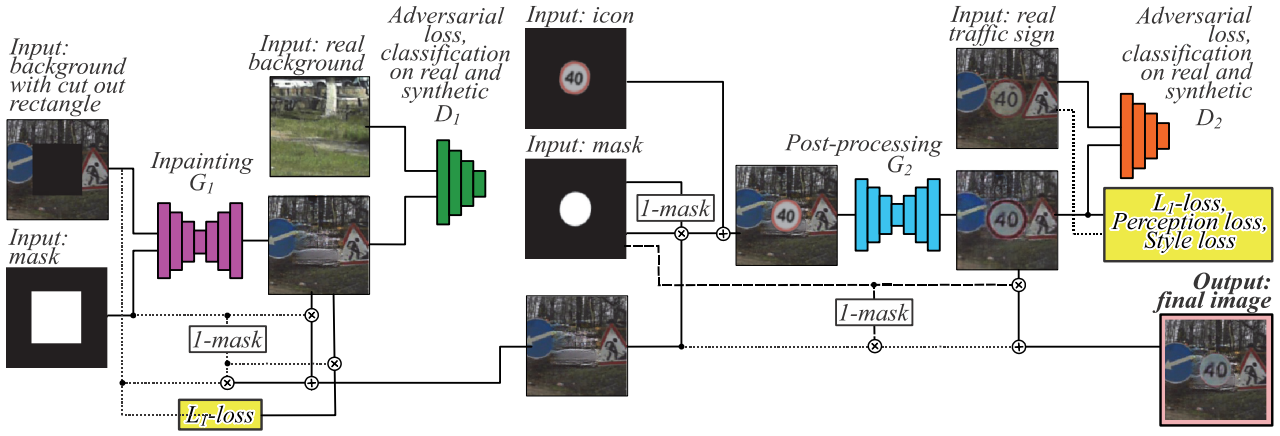


*Fig. 3. Additional data route in second approach*

## 2.2. Location of embedded traffic signs in real road images

We have also examined the geometric positioning of traffic signs in the image. We train a neural network that will find appropriate places for additional traffic signs on road images.

We have used sampling with kernel density estimation from the distribution of existing labelled data positions as a baseline for the placement of additional traffic signs. This approach does not take into account the features of each particular image. It was trained on real labelled training samples. We built three different distributions – to sample coor-

dinates of the signs in the image, sample their sizes and the number of signs in the current image.

Next, we have tried the approach from [4]. In our work, we have used the only *where* module, while *what* module was disabled. We didn't find any papers, where such type of neural networks was used for traffic sign placement previously.

For the given image, the model tries to predict the correct distribution of sizes and locations of object instances. It is a generative adversarial network, where the generator as input takes semantic segmentation of image and a random vector. As output, it returns parameters of affine transformation without rotation for an appropriate bounding box for a new sign.

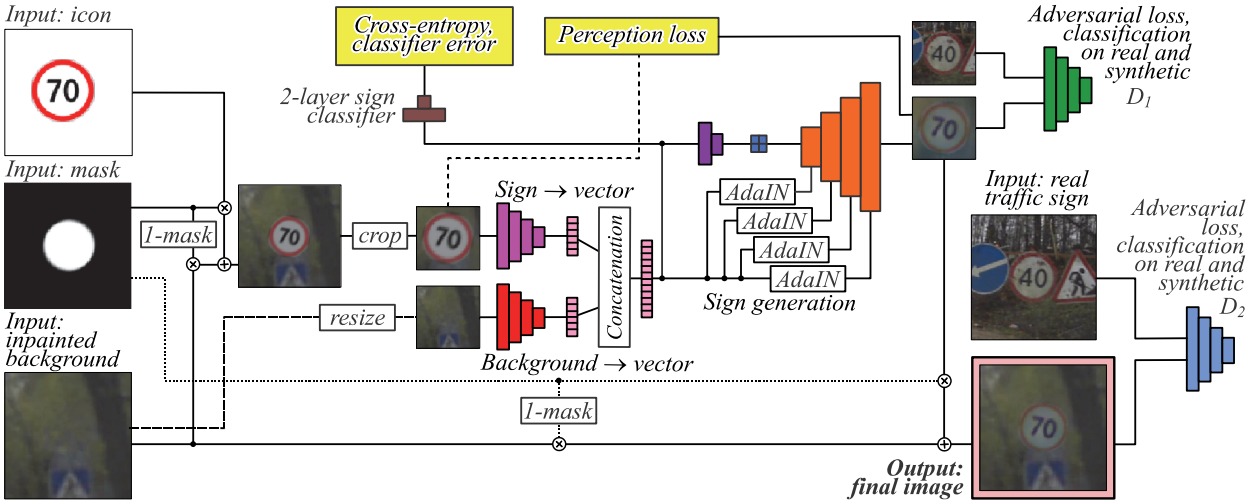


*Fig. 4. Architecture of generator for processing in third approach*

This architecture has two discriminators. First, D1 learns to differentiate real and generated affine parameters for the current image. Second, D2 learns to distinguish whether a new bounding object is consistent with the input semantic map. Cross-entropy is used as an adversarial loss for them. During training, this module has two paths – unsupervised and supervised. An unsupervised path has only a second discriminator D2, while a supervised has both D1 and D2.

For unsupervised path, architecture has input reconstruction loss, which aims to reconstruct input semantic map and random vector from intrinsic representation for STN subnetwork using *L1-loss*. This helps to ensure that encoded representation has significant information from input data and partially solves the problem of model collapsing to a few numbers of modes and not covering the entire distribution.

In the supervision path, we already have one of the real positions of traffic signs. This information should be conveyed to architecture. To achieve this, the network has an additional submodule that encodes real affine transformation to the input vector (instead of random) and output transformation should be the same as the input. Kullback-Leibler divergence term in loss helps this submodule to learn the correct distribution for encoding. D1 tries to distinguish synthetic parameters. This path also helps positions determined by the transform to become more diverse.

This neural network for determining the location of objects is based on semantic maps. Since the RTSD dataset does not have semantic segmentation, we first conducted experiments, in which RGB road images are fed to the input of a neural network. With such training, we were not able to achieve acceptable quality and the generated distributions themselves collapsed into degenerate when all new signs are located in the same place for every image.

To solve the problem of missing RTSD semantic segmentation, we have applied to our dataset the semantic segmentation model, trained on Cityscapes dataset. We have used the pre-trained method 'Fast Semantic Segmentation' [39]. It generates plausible semantic segmentation. We have used the obtained semantic maps to train *where* module of the neural network for object placement.

After that, we have used a trained neural network to sample the locations and sizes of new traffic signs. When generating them, we have made sure that the new examples did not overlap. The number of traffic signs for each image has been determined using Gaussian kernel density estimation. A full pipeline of new traffic sign generation process is portrayed in fig. 5.

## 3. Evaluation

### 3.1. RTSD Dataset

As real data, we took Russian traffic sign dataset RTSD [40]. It consists of 205 classes, of which 99 are found only in the test set and are completely absent in the training set, and 106 classes are present in the training set. The set contains data for training detectors and classifiers of traffic signs. Train and test data statistics can be found in tables 1, 2.

*Table 1. Statistics for detection task RTSD dataset*

| | Images | Signs |
|---|---|---|
| Train set, Pasted, Cycled, Styled | 47639 | 80277 |
| Test set | 11389 | 25232 |

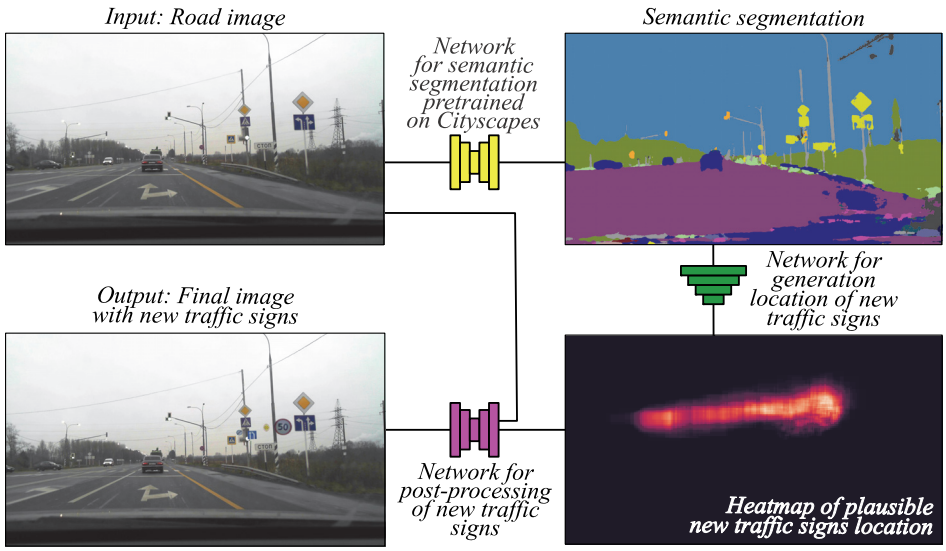


*Fig. 5. A full pipeline of proposed traffic sign placement method*


*Fig. 6. Comparison of different synthetic traffic signs types*

*Table 2. Statistics for classification task RTSD dataset*

| | All | Rare | Frequent |
|---|---|---|---|
| Train set | 79896 | 0 | 79896 |
| Test set | 25613 | 1622 | 23991 |
| CGI-GAN | 193444 | 94465 | 98979 |
| Pasted, Cycled, Styled | 196455 | 94472 | 101983 |

Also for all 205 classes, we had high-resolution icons of traffic signs with their masks.

We compared our proposed approach for embedding synthetic objects in pictures with three already existed methods for traffic sign [28] processing:

- **Synt** – this is a simple synthetic, which was obtained by embedding signs on the background and applying a transformation of sign with random parameters to the icon: rotate, shift, contrast change, Gaussian blur, motion blur.
- **CGI** – samples, which were obtained by rendering three-dimensional models of traffic signs on pillars in real road images.
- **CGI-GAN** – in this sample, traffic signs are transformed from the **CGI** collection to better ones using CycleGAN.
- **Inpaint** – this is a simple synthetic data for the detector, in which an icon of a traffic sign is drawn in the image without any processing.

### *3.2. Generated data sets*

To begin with, we conducted experiments, in which synthetic traffic signs were embedded in places where real ones already existed. This secured the correct geometric placement of synthetic signs. For detector, the number of images and signs in the synthetic set is the same as in the real dataset. For classifier, the number of samples is the same as for previously existed synthetic data sets. Let's introduce abbreviations for the proposed three models:

- **Pasted** – results of first approach.
- **Cycled** – results of second approach.
- **Styled** – results of third approach.

Then we generated a synthetic set for the detector, in which new places of traffic signs were determined or using kernel density estimation (**KDE**) or using a special neural network (**NN**). At the same time, the processing of synthetic signs was made only by the third **Styled** approach, which showed itself best for rare traffic signs. In this method, we conducted various experiments. Let's introduce abbreviations for proposed approaches:

- **KDE-additional** and **NN-additional** – Locate new traffic signs in addition to existing ones.
- **KDE-only-synt** and **NN-only_synt** – Perform inpainting of real signs to remove them, and then place synthetic signs in new places. As a result, there are no signs in such pictures at the places of previously existed.
- **KDE-manystyled** and **NN-manystyled** – Perform inpainting of real signs, and then place synthetic signs in both new places and existed in real places.

The number of images in each set was the same as the number of images in the training set, because each training image was augmented exactly one time.

### *3.3. Traffic sign recognition system*

As an object detector, we use PVANet [41], which is based on the Faster R-CNN approach. We evaluated detection output on a test set before and after we applied the classifier. The area under curve (AUC) was used to measure detector quality.

As traffic sign classifiers we chose two models based on WideResNet [36]. The first one is a simple classifier model with WideResNet architecture. It takes an image of size $64 \times 64$ pixels and predicts one of the 205 sign classes. On the features extracted by this neural network, we trained a simple k-NN classifier. It operates on an index that consists of synthetic examples of traffic signs. The second method is designed specifically to handle the case of rare traffic sign classes. It is proposed in paper [35]. In this method, rare and frequent classes are treated differently. First, WideResNet features are extracted at a penultimate layer of the neural network. These features are then used in Random Forest to classify whether a sign is rare or frequent. Frequent signs are classified with the Softmax layer on top of WideResNet. Rare classes are passed into a k-NN classifier. This classifier shows better quality compared to the first classifier [35]. To measure quality, we first calculated overall accuracy on the test set. In the same tables, separately for rare and frequent classes, we calculated the micro-averaged Recall (formula 1), as it is important for us to understand how many signs we find from the available ones. M is the number of classes. For class $i$ we define $TP_i$, $FN_i$, $FP_i$ as the number of true positives, false negatives and false positives respectively.

$$\text{micro-averaged Recall} = \frac{\sum_{i=1}^{M} TP_i}{\sum_{i=1}^{M} (TP_i + FN_i)}. \quad (1)$$

Next, we compared the macro-averaged Precision(2), Recall (3), and F1 (4) measures for all classes and separately for rare and frequent classes.

$$\text{macro-averaged Precision} = \frac{\sum_{i=1}^{M} \frac{TP_i}{TP_i + FP_i}}{M}, \quad (2)$$

$$\text{macro-averaged Recall} = \frac{\sum_{i=1}^{M} \frac{TP_i}{TP_i + FN_i}}{M}, \quad (3)$$

$$\text{macro-averaged F1} = \frac{\sum_{i=1}^{M} \frac{2*\text{Precision}_i * \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}}{M}. \quad (4)$$

*real image* *image with replaced sings* *image with additional signs*

*Fig. 7. Comparison of real images; images from **Styled** set, where real images replaced with synthetic ones; images from **NN-additional** set, where new traffic signs were located in addition to existing ones*

### *4. Evaluation results*

During our experiments, we trained neural networks for classification and detection both on mixtures of real data with synthetic data and on synthetic data alone. A comparison of traffic signs examples for a classifier can be seen in table 6. Examples of road images with synthetic signs are in table 7.

#### *4.1. Classifier results*

Here we describe the results of experiments with classifiers. We compared the proposed approach with the best previous method, which uses synthetic data **CGI-GAN** [35]. Table 3 shows measurements of a simple WideResNet classifier trained on a mixture of real and synthetic samples with a k-NN index trained on its features. For all classes we measured accuracy, but separately for rare and frequent classes we measured micro-average recall. Table 4 shows measurements of a simple WideResNet classifier, trained only on synthetic samples. Table 5 summarizes measurements of improved WideResNet classifier trained on a mixture of real and synthetic samples. Table 6 shows metrics of an improved WideResNet classifier trained only on synthetic samples. Table 8 shows macro-averaged Precision, Recall, and F1 measures for WideResNet classifiers trained only on synthetic samples. And table 7 for classifiers trained on a mixture of real and synthetic samples.

The best results are highlighted in tables. Obtained values show that approaches **Cycled** and **Styled** compete in terms of quality for target classifiers. It's hard to say, which data is better. It depends on the specific task in which target classifiers will be used.

*Table 3. Simple WideResNet classifier trained on a mixture of real and synthetic samples with a k-NN index on its features*

| | Metrics of softmax output | | | Metrics with index from icons | | | Metrics with index from test set | | | Metrics with index from synthetic set | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | all, Acc | rare, Recall | frequent, Recall | all, Acc | rare, Recall | frequent, Recall | all, Acc | rare, Recall | frequent, Recall | all, Acc | rare, Recall | frequent, Recall |
| RTSD | 88.87 | 0.00 | 94.88 | 71.53 | 41.68 | 73.55 | 73.00 | 61.13 | 73.75 | | | |
| RTSD + CGI-GAN | 92.75 | 53.82 | **95.39** | 77.03 | 61.34 | 78.09 | 74.83 | 65.79 | 75.41 | 82.15 | 59.62 | 83.67 |
| RTSD + Pasted | 91.67 | 68.74 | 93.22 | 76.50 | 68.74 | 77.02 | 76.34 | **73.34** | 76.53 | 85.42 | 71.33 | 86.37 |
| RTSD + Cycled | 92.03 | 68.19 | 93.64 | **78.59** | **72.19** | **79.03** | **78.12** | 70.65 | **78.60** | **86.71** | **73.43** | **87.61** |
| RTSD + Styled | **92.82** | **69.67** | 94.39 | 76.19 | 69.05 | 76.67 | 77.8 | 69.27 | 78.35 | 85.96 | 70.41 | 87.01 |

*Table 4. Simple WideResNet classifier trained only on synthetic samples*

| | Metrics of softmax output | | | Metrics with index from icons | | | Metrics with index from test set | | | Metrics with index from synthetic set | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | all, Acc | rare, Recall | frequent, Recall | all, Acc | rare, Recall | frequent, Recall | all, Acc | rare, Recall | frequent, Recall | all, Acc | rare, Recall | frequent, Recall |
| only CGI-GAN | 49.80 | 43.90 | 50.19 | 22.88 | 23.30 | 22.85 | 44.10 | 46.09 | 43.97 | 39.35 | 34.09 | 39.70 |
| only Pasted | 67.5 | 69.05 | 67.39 | 49.87 | 47.78 | 50.01 | 58.50 | 60.93 | 58.34 | 66.47 | 65.66 | 66.52 |
| only Cycled | **73.61** | 65.60 | **74.15** | **64.05** | **54.69** | **64.69** | 60.01 | 62.97 | 59.82 | **72.65** | 63.69 | **73.26** |
| only Styled | 69.94 | **69.42** | 69.97 | 42.68 | 48.27 | 42.30 | **65.72** | **69.34** | **65.49** | 71.77 | **67.39** | 72.06 |

*Table 5. Improved WideResNet classifier trained on a mixture of real and synthetic samples*

| | Metrics with index from icons | | | Metrics with index from test set | | | Metrics with index from synthetic set | | |
|---|---|---|---|---|---|---|---|---|---|
| | all, Accuracy | rare, Recall | frequent, Recall | all, Accuracy | rare, Recall | frequent, Recall | all, Accuracy | rare, Recall | frequent, Recall |
| RTSD + CGI-GAN | 93.12 | 70.65 | 94.43 | 92.11 | 76.69 | 93.10 | 93.52 | 70.16 | 95.09 |
| RTSD + Pasted | 92.75 | 73.80 | 94.03 | 92.86 | **81.42** | 93.59 | 93.84 | 74.97 | 95.11 |
| RTSD + Cycled | **93.31** | **76.70** | **94.44** | **93.24** | 79.65 | **94.11** | 93.98 | 75.46 | 95.23 |
| RTSD + Styled | 92.16 | 75.83 | 93.26 | 93.04 | 78.92 | 93.94 | **94.11** | **76.33** | **95.31** |

*Table 6. Improved WideResNet classifier trained only on synthetic samples*

| | Metrics with index from icons | | | Metrics with index from test set | | | Metrics with index from synthetic set | | |
|---|---|---|---|---|---|---|---|---|---|
| | all, Accuracy | rare, Recall | frequent, Recall | all, Accuracy | rare, Recall | frequent, Recall | all, Accuracy | rare, Recall | frequent, Recall |
| only CGI-GAN | 59.68 | 57.77 | 59.81 | 59.58 | 66.84 | 59.12 | 60.55 | 54.69 | 60.94 |
| only Pasted | 71.51 | **73.06** | 71.40 | 70.77 | **78.00** | 70.31 | 72.57 | 72.13 | 72.60 |
| only Cycled | **72.41** | 70.59 | **72.54** | 72.18 | 77.28 | 71.84 | 72.35 | 71.89 | 72.38 |
| only Styled | 71.82 | 71.15 | 71.86 | **73.84** | 77.54 | **73.60** | **73.03** | **73.98** | **72.96** |

*Table 7. Macro-averaged Precision, Recall and F1 for WideResNet classifiers trained on a mixture of real and synthetic data*

| | Metrics of softmax output | | | | | | | | | Metrics with index from synthetic set | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | all | | | rare | | | frequent | | | all | | | rare | | | frequent | | |
| | precision | recall | F1 | precision | recall | F1 | precision | recall | F1 | precision | recall | F1 | precision | recall | F1 | precision | recall | F1 |
| Simple WideResNet classifier | | | | | | | | | | | | | | | | | | |
| RTSD + CGI-GAN | 69.98 | 70.27 | 67.80 | 51.11 | 52.06 | 47.80 | 87.6 | 87.28 | 86.48 | 60.94 | 69.03 | 60.68 | 41.22 | 57.29 | 42.05 | 79.36 | 80.00 | 78.07 |
| RTSD + Pasted | 72.79 | 75.88 | 71.83 | 53.95 | 60.06 | 52.22 | 90.39 | 90.65 | 90.14 | 62.22 | 75.92 | 65.09 | 40.02 | 64.24 | 44.94 | 82.95 | 86.83 | 83.91 |
| RTSD + Cycled | 74.01 | 75.12 | 72.01 | 56.78 | 58.32 | 52.76 | 90.11 | 90.8 | 89.99 | **64.38** | 74.44 | 66.11 | **43.28** | 61.22 | **46.48** | 84.09 | 86.79 | 84.44 |
| RTSD + Styled | **76.11** | **76.67** | **74.11** | **59.45** | **60.31** | **55.60** | **91.67** | **91.95** | **91.38** | 63.65 | **76.56** | **66.4** | 40.65 | **64.46** | 45.84 | **85.14** | **87.86** | **85.60** |
| Improved WideResNet classifier | | | | | | | | | | | | | | | | | | |
| RTSD + CGI-GAN | | | | | | | | | | 73.88 | 76.34 | 72.38 | 54.64 | 59.84 | 52.07 | 91.84 | 91.75 | 91.34 |
| RTSD + Pasted | | | | | | | | | | 74.38 | 78.93 | 74.38 | 54.26 | 65.06 | 55.30 | 93.16 | 91.89 | 92.20 |
| RTSD + Cycled | | | | | | | | | | 76.51 | 79.65 | 75.59 | **58.75** | **67.78** | **58.73** | 93.10 | 90.74 | 91.33 |
| RTSD + Styled | | | | | | | | | | **76.69** | **80.12** | **76.24** | **58.74** | 66.43 | 58.41 | **93.45** | **92.9** | **92.89** |

*Table 8. Macro-averaged Precision, Recall and F1 measures for WideResNet classifiers trained only on synthetic samples*

| | Metrics of softmax output | | | | | | | | | Metrics with index from synthetic set | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | all | | | rare | | | frequent | | | all | | | rare | | | frequent | | |
| | precision | recall | F1 | precision | recall | F1 | precision | recall | F1 | precision | recall | F1 | precision | recall | F1 | precision | recall | F1 |
| Simple WideResNet classifier | | | | | | | | | | | | | | | | | | |
| RTSD + CGI-GAN | 40.45 | 46.19 | 38.15 | 22.30 | 39.96 | 24.32 | 57.40 | 52.01 | 51.07 | 36.56 | 37.52 | 31.72 | 17.71 | 31.30 | 18.27 | 54.17 | 43.34 | 44.27 |
| RTSD + Pasted | 49.11 | 66.45 | 50.82 | 28.35 | **61.43** | 33.79 | 68.49 | 71.14 | 66.73 | 45.88 | 63.85 | 47.47 | 23.14 | 58.55 | 29.42 | 67.11 | 68.80 | 64.33 |
| RTSD + Cycled | 51.95 | 65.27 | 53.29 | **31.40** | 56.33 | **35.73** | 71.15 | 73.62 | 69.69 | 49.71 | 66.33 | 51.81 | 28.10 | **59.23** | 33.39 | 69.90 | 72.95 | 69.01 |
| RTSD + Styled | **53.31** | **68.97** | **54.96** | 30.51 | 59.86 | **35.72** | **74.60** | **77.48** | **72.93** | **53.70** | **68.8** | **56.14** | **30.95** | 58.57 | **36.62** | **74.95** | **78.36** | **74.37** |
| Improved WideResNet classifier | | | | | | | | | | | | | | | | | | |
| RTSD + CGI-GAN | | | | | | | | | | 45.73 | 55.96 | 45.42 | 26.32 | 48.38 | 29.94 | 63.85 | 63.04 | 59.87 |
| RTSD + Pasted | | | | | | | | | | 55.60 | 73.37 | 58.46 | 35.80 | 66.02 | 41.67 | 74.08 | 80.24 | 74.15 |
| RTSD + Cycled | | | | | | | | | | 56.80 | 69.80 | 58.65 | 36.75 | 60.69 | 41.42 | 75.53 | 78.31 | 74.74 |
| RTSD + Styled | | | | | | | | | | **58.79** | **74.00** | **61.03** | **39.13** | **66.38** | **43.69** | **77.15** | **81.12** | **77.23** |

However, 94.11 % is the best accuracy value that was obtained by classifying both rare and frequent classes with an improved classifier (table 5). It was achieved by training using the **Styled** data and classification by index from corresponding synthetic data. Previously, the best quality was 93.52 %. Micro-average recall of rare traffic signs has also greatly improved from 70.16 to 76.33. Table 7 also shows that we were able to improve macro-averaged precision, recall and F1 for the simple and improved WideResNet classifiers using the proposed synthetic data in comparison with the **CGI-GAN**. The best results were shown by methods **Cycled** and **Styled**. For all classes, F1-measure has grown from 72.38 to 76.24 with **Styled**, for rare from 52.07 to 58.73 with **Cycled**, and for frequent ones from 91.34 to 92.89 with **Styled**.

Improved classifier still allows obtaining better quality compared to the usual one (tables 5, 7). For usual maximum accuracy, the value is 92.82 % (this is less than 94.11 % for the improved one). Macro-averaged precision, recall and F1 also better with improved classifier than with usual. This once again confirms the assumption of previous article [35].

It is also seen that proposed synthetic data significantly improve the quality of classification when training only on synthetic data. Previously, the best result was 60.55 %, and now 73.03 % (table 6).

Therefore we conclude that usage of proposed synthetic samples during training the process allows improving the quality of the WideResNet classifier.

### *4.2. Detector results*

Next, we present the results of experiments with a detector. Table 9 shows AUC values for a detector trained on a mixture of real and synthetic samples. The table 10 shows AUC measurements for a detector trained only on synthetic samples.

The best results are highlighted in tables. It can be seen that without intelligent placement with the neural network it is not possible to improve the detection quality of frequent classes by any synthetic data. It is 89.25 for frequent classes when training only on real data. The best quality using synthetic samples without neural network placement is achieved at **Styled** and is equal to 89.13. On rare signs, AUC increases from 85.86 to 86.78. It can be seen from the table that proposed approaches for post-processing of synthetic signs work better than already existed. At the same time, we improved the quality of sequential detection and classification of traffic signs due to classifiers' improvement. Our average AUC increased from 86.01 to 86.11. On rare signs, the increase is more significant – from 58.56 to 64.20.

The best results are obtained if VAE-based neural network is used for placement of new signs. Gaussian kernel density estimation works a bit worse. That means that the proposed method for location generation with a neural network is better for synthetic training data than random placement with samples generated from distribution. Without a classifier, we achieved 89.17 average AUC and 89.31 for frequent signs, which is the best result. For rare classes metric is a bit worse (86.62) than achieved with **Styled** 86.78. When using **Styled** classifier this method has the best results for average and for rare / frequent. Thus our best average AUC is 86.16.

Training only on synthetic data shows that the synthetic set closest to real data is **Styled**. It turns out to achieve a significantly higher quality of detection in comparison with other synthetic sets. On average it equals to 62.12 without a classifier and 39.99 with a classifier.

## *Conclusion*

In this paper, we proposed a neural network-based method for embedding new synthetic traffic signs in road

images. The proposed method consists of networks for the placement and processing of new signs. For placement, we use a neural network with Spatial Transformer Network to predict the best locations for additional signs. Also, we developed three improved models for the processing of traffic signs. The idea of the proposed approaches is to use neural networks consisting of two parts. One is used to inpaint traffic signs that have been already located in the image, and the second is used to embed new ones. It the two of the proposed methods these two parts are being trained jointly, but in the third approach separately.

*Table 9. Detector trained on a mixture of a real and synthetic samples*

| | Without classifier | | | With classifier | | |
|---|---|---|---|---|---|---|
| | all | rare | frequent | all | rare | frequent |
| RTSD | 89.09 | 85.86 | 89.25 | 86.01 | 58.56 | 86.61 |
| RTSD + CGI | 88.56 | 85.72 | 88.72 | 83.84 | 48.51 | 85.15 |
| RTSD + Inpaint | 88.61 | 86.63 | 88.71 | 76.41 | 34.00 | 82.93 |
| RTSD + Pasted | 88.98 | 86.59 | 89.09 | 85.81 | 59.98 | 86.40 |
| RTSD + Cycled | 88.98 | 86.29 | 89.13 | 86.11 | 60.13 | 86.62 |
| RTSD + Styled | 89.01 | **86.78** | 89.13 | 85.39 | 64.20 | 86.13 |
| RTSD + KDE-only-synt | 89.03 | 86.34 | 89.17 | 85.59 | 63.83 | 86.28 |
| RTSD + NN-only-synt | 88.79 | 86.34 | 88.92 | 85.43 | 64.76 | 86.11 |
| RTSD + KDE-manystyled | 88.88 | 86.32 | 89.01 | 85.34 | 62.70 | 86.01 |
| RTSD + NN-manystyled | 88.95 | 86.18 | 89.09 | 85.37 | 63.65 | 86.07 |
| RTSD + KDE-additional | 89.11 | 86.52 | 89.22 | 85.99 | 64.83 | 86.54 |
| RTSD + NN-additional | **89.17** | 86.62 | **89.31** | **86.16** | **64.96** | **86.70** |

*Table 10. Detector trained on synthetic samples*

| | Without classifier | | | With classifier | | |
|---|---|---|---|---|---|---|
| | all | rare | frequent | all | rare | frequent |
| only CGI | 10.70 | 13.23 | 10.63 | 8.81 | 8.53 | 8.81 |
| only Inpaint | 55.23 | **55.26** | 56.26 | 15.89 | 13.79 | 15.96 |
| only Pasted | 38.22 | 38.97 | 38.17 | 19.85 | 18.80 | 19.91 |
| only Cycled | 37.20 | 42.97 | 37.13 | 25.88 | 24.42 | 25.94 |
| only Styled | **62.12** | 54.26 | **62.36** | **39.99** | **32.51** | **40.35** |
| only KDE-only-synt | 51.22 | 42.76 | 51.61 | 32.36 | 32.68 | 25.83 |
| only NN-only-synt | 50.49 | 42.02 | 50.83 | 31.88 | 25.52 | 32.19 |
| only KDE-manystyled | 60.39 | 52.99 | 60.68 | 39.09 | 32.59 | 39.44 |
| only NN-manystyled | 60.33 | 52.57 | 60.64 | 38.66 | 31.63 | 39.04 |

We also made a comparison of different synthetic data quality by training target neural networks. It showed that our method for generating synthetic training samples improved quality for detector and classifier of traffic signs. For all sign classes, recognition quality improved. The most noticeable improvement was achieved for rare classes, which absolutely absent in the original real training set.

## *References*


[1] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. NIPS'14 Proc 27th Int Conf on Neural Information Processing Systems 2014; 2: 2672-2680.

[2] Zhu J-Y, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. Proc IEEE Int Conf on Computer Vision 2017: 2223-2232.

[3] Karras T, Laine S, Aila T. A style-based generator architecture for generative adversarial networks. Proc IEEE Conf on Computer Vision and Pattern Recognition 2019: 4401-4410.

[4] Lee D, Liu S, Gu J, Liu M-Y, Yang M-H, Kautz J. Context-aware synthesis and placement of object instances. NIPS'18 Proc 32nd Int Conf on Neural Information Processing Systems 2018: 10414-10424.

[5] Dwibedi D, Misra I, Hebert M. Cut, paste and learn: Surprisingly easy synthesis for instance detection. Proc IEEE Int Conf on Computer Vision 2017: 1301-1310.

[6] Dvornik N, Mairal J, Schmid C. Modeling visual context is key to augmenting object detection datasets. Proc European Conf on Computer Vision (ECCV) 2018: 364-380.

[7] Zhang S, Liang R, Wang M. Shadowgan: Shadow synthesis for virtual objects with conditional adversarial networks. Comput Vis Media (Beijing) 2019; 5(1): 105-115.

[8] Liu L, Muelly M, Deng J, Pfister T, Li L-J. Generative modeling for small-data object detection. Proc IEEE Int Conf on Computer Vision 2019: 6073-6081.

[9] Reed A, Gerg ID, McKay JD, Brown DC, Williamsk DP, Jayasuriya S. Coupling rendering and generative adversarial networks for artificial sas image generation. OCEANS 2019 MTS/IEEE SEATTLE 2019: 1-10.

[10] Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434. Source: ⟨https://arxiv.org/abs/1511.06434⟩.

[11] Denton EL, Chintala S, Fergus R, et al. Deep generative image models using a laplacian pyramid of adversarial networks. NIPS'15 Proc 28th Int Conf on Neural Information Processing Systems 2015; 1: 1486-1494⟩.

[12] Mirza M, Osindero S. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784. Source: ⟨https://arxiv.org/abs/1411.1784.

[13] Huang X, Belongie S. Arbitrary style transfer in realtime with adaptive instance normalization. Proc IEEE Int Conf on Computer Vision 2017: 1501-1510.

[14] Chen B-C, Kae A. Toward realistic image compositing with adversarial learning. Proc IEEE Conf on Computer Vision and Pattern Recognition 2019: 8415-8424.

[15] Zheng Z, Zheng L, Yang Y. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. Proc IEEE Int Conf on Computer Vision 2017: 3754-3762.
[16] Frid-Adar M, Klang E, Amitai M, Goldberger J, Greenspan H. Synthetic data augmentation using gan for improved liver lesion classification. IEEE 15th Int Symposium on Biomedical Imaging (ISBI 2018) 2018: 289-293.
[17] Richter SR, Vineet V, Roth S, Koltun V. Playing for data: Ground truth from computer games. European Conference on Computer Vision 2016: 102-118.
[18] Gaidon A, Wang Q, Cabon Y, Vig E. Virtual worlds as proxy for multi-object tracking analysis. Proc IEEE Conf on Computer Vision and Pattern Recognition 2016: 4340-4349.
[19] Hoffman J, Tzeng E, Park T, Zhu J-Y, Isola P, Saenko K, Efros A, Darrell T. Cycada: Cycle-consistent adversarial domain adaptation. Int Conf on Machine Learning 2018: 1989-1998.
[20] Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B. The cityscapes dataset for semantic urban scene understanding. Proc IEEE Conf on Computer Vision and Pattern Recognition 2016: 3213-3223.
[21] Jaderberg M, Simonyan K, Zisserman A, et al. Spatial transformer networks. NIPS'15 Proc 28th Int Conf on Neural Information Processing Systems 2015: 2017-2025.
[22] Tripathi S, Chandra S, Agrawal A, Tyagi A, Rehg JM, Chari V. Learning to generate synthetic data via compositing. Proc IEEE Conf on Computer Vision and Pattern Recognition 2019: 461-470.
[23] De La Escalera A, Moreno LE, Salichs MA, Armingol JM. Road traffic sign detection and classification. IEEE Trans Ind Electron 1997; 44(6): 848-859.
[24] Hoessler H, Wohler C, Lindner F, Kreßel U. Classifier training based on synthetically generated samples. Proc 5th Int Conf on Computer Vision Systems (ICVS) 2007. Source: ⟨https://biecoll.ub.uni-bielefeld.de/index.php/icvs/article/view/201/294⟩.
[25] Chigorin A, Konushin A. A system for large-scale automatic traffic sign recognition and mapping. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 2013; II-3/W3: 13-17.
[26] Moiseyev B, Konev A, Chigorin A, Konushin A. Evaluation of traffic sign recognition methods trained on synthetically generated data. In Book: Blanc-Talon J, Kasinski A, Philips W, Popescu D, Scheunders P, eds. Advanced concepts for intelligent vision systems. Cham, Heidelberg: Springer; 2013: 576-583.
[27] Zhu Z, Liang D, Zhang S, Huang X, Li B, Hu S. Traffic-sign detection and classification in the wild. Proc IEEE Conf on Computer Vision and Pattern Recognition 2016: 2110-2118.
[28] Shakhuro V, Faizov B, Konushin A. Rare traffic sign recognition using synthetic training data. Proc 3rd Int Conf on Video and Image Processing (ICVIP) 2019: 23-26.
[29] Liang Z, Shao J, Zhang D, Gao L. Traffic sign detection and recognition based on pyramidal convolutional networks. Neural Comput Appl 2020; 32: 6533-6543.
[30] Ayachi R, Afif M, Said Y, Atri M. Traffic signs detection for real-world application of an advanced driving assisting system using deep learning. Neural Process Lett 2020; 51(1): 837-851.
[31] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM 2017; 60(6): 84-90.
[32] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Proc IEEE Conf on Computer Vision and Pattern Recognition 2016: 770-778.
[33] Farag W. Traffic signs classification by deep learning for advanced driving assistance systems. Intell Decis Technol 2019; 13(3): 305-314.
[34] Serna CG, Ruichek Y. Classification of traffic signs: The european dataset. IEEE Access 2018; 6: 78136-78148.
[35] Faizov B, Shakhuro V, Sanzharov V, Konushin A. Classification of rare traffic signs. Computer Optics 2020; 44(2): 236-243. DOI: 10.18287/2412-6179-CO-601.
[36] Zagoruyko S, Komodakis N. Wide residual networks. arXiv preprint arXiv:1605.07146. Source: ⟨https://arxiv.org/abs/1605.07146⟩.
[37] Nazeri K, Ng E, Joseph T, Qureshi FZ, Ebrahimi M. EdgeConnect: Generative image inpainting with adversarial edge learning. arXiv preprint arXiv:1901.00212. Source: ⟨https://arxiv.org/abs/1901.00212⟩.
[38] Johnson J, Alahi A, Fei-Fei L. Perceptual losses for real-time style transfer and super-resolution. In Book: Leibe B, Matas J, Sebe N, Welling M, eds. Computer Vision – ECCV 2016. Springer International Publishing AG; 2016: 694-711.
[39] Andrienko O. Fast semantic segmentation. Source: ⟨https://github.com/oandrienko/fast-semantic-segmentation⟩.
[40] Shakhuro V, Konushin A. Russian traffic sign images dataset. Computer Optics 2016; 40(2): 294-300. DOI: 10.18287/2412-6179-2016-40-2-294-300.
[41] Kim K-H, Hong S, Roh B, Cheon Y, Park M. PVANet: Deep but lightweight neural networks for real-time object detection. arXiv preprint arXiv:1608.08021. Source: ⟨https://arxiv.org/abs/1608.08021⟩.

### *Authors' information*

**Anton Sergeevich Konushin** (b. 1980), graduated from Lomonosov Moscow State University in 2002. In 2005 he successfully defended his PhD thesis in M.V. Keldysh Institute for Applied Mathematics RAS. He is currently associate professor at Lomonosov Moscow State University. Research interests are computer vision and machine learning. E-mail: *anton.konushin@graphics.cs.msu.ru* .

**Boris Vladimirovich Faizov** (b. 1998), student at Lomonosov Moscow State University. Research interests are computer vision, machine learning and programming. E-mail: *boris.faizov@graphics.cs.msu.ru* .

**Vladislav Igorevich Shakhuro** (b. 1993), graduated from Lomonosov Moscow State University in 2015. Research interests are image processing, computer vision, machine learning, and programming.
E-mail: *vlad.shakhuro@graphics.cs.msu.ru* .

*Received December 30, 2020. The final version – April 19, 2021.*