GANERaTE: GAN Empowered Realistic Synthetic Traffic Sign Enhancement

Andrei-Daniel Bălănici, Andrei-Robert Alexandrescu, Alexandru Manole
Department of Computer-Science
Faculty of Mathematics and Computer Science
Babeș-Bolyai University
Cluj-Napoca, Romania
andrei.balanici2001@gmail.com, andrei.robert.alexandrescu@gmail.com, alexandru.manole@stud.ubbcluj.ro

***Abstract*—In domains where annotated examples of objects of interest are scarce, the usage of synthetic objects has exhibited improved performance. Generative Adversarial Networks (GANs) have shown significant advancements in solving computer vision tasks, encompassing image generation, image-to-image translation, and style transfer. This research paper explores the application of GANs to the synthesis of traffic signs, employing a two-step approach to achieve realistic and diverse results. Through experimentation on multiple datasets, we evaluate the efficacy of synthetic traffic signs by training a YOLO v5 model. The findings indicate that the synthetic signs' enhancement improves the performance of the detector by 3%.**

***Index Terms*—Image Generation, Synthetic Data, Traffic Signs, Object Detection**

I. Introduction

Generating synthetic images in the computer vision domain represents an essential task toward enlarging the corpus of annotated data. Synthetic data is useful especially when natural, real-world data is difficult to collect and annotate. Often times, it is cheaper to generate images and use them to train machine learning models than to gather real data from the environment. The trade-offs between the two options of data collection are similar to the trade-offs between creating real special effects for movies compared to synthetic effects made with computer software. The former is real and qualitative, but expensive and unpractical, while the latter usually leads to worse, less real results, but at a fraction of the cost.

One of the fields where synthetic data would have a large impact is the domain of scene understanding for autonomous vehicles. When developing an autonomous system for environment understanding, the engineers must ensure that the system is robust enough to take the right decisions in a diverse set of scenarios. Autonomous vehicles meant to work on public roads must be able to comprehend all possible traffic signs as concisely as possible and take decisions based on their placement. The difficulty of this task arises from the lack of availability of real data on rare traffic signs. By rare, we mean traffic signs which are harder to find in real scenarios, therefore underrepresented in the dataset or even completely missing as is the case of the retractable bridge traffic sign.

Instead of investing large efforts towards gathering real footage of rare signs, synthetic image generation represents a viable option. The intent of this approach is to generate realistic rare traffic signs and blend them into real scenarios. This is a challenging task because in order to train a model to generate realistic signs, extensive amounts of data with those signs are required. For this reason, researchers must find a way to generate such signs with as few training data samples as possible.

In this article, we study the problem of synthetic traffic sign generation in ego-view images from the perspective of an autonomous car. We attempt to replicate and improve an existing method proposed in [8]. The task is split into two parts: inpainting of background and realism enhancement. Both tasks are solved using variants of Generative Adversarial Networks [4]. The evaluation of the method is done by adding synthetic signs in images and comparing the performance of the YOLO object detector [15] on the real and synthetic datasets.

Our research aims to reveal meaningful answers to the following research questions:

- Can we improve the performance of an object detector, in our case YOLO, using synthetic images?
- How many synthetic signs are necessary to improve the performance of object detectors?

The paper is structured as follows. Section II introduces related concepts and works from the field and Section III shapes the methodology used in our work for realistic traffic sign generation. The visual and numerical results are given in Section IV, together with an analysis of them. Section V concludes this article, providing future considerations.

II. Concepts

A. Generative Adversarial Networks

Generative Adversarial Networks (GANs) were first introduced by Godfellow et al. in 2014 as a novel framework for estimating generative models via an adversarial process [4]. This process implies training two artificial neural networks at the same time. One of the networks, named Discriminator, is responsible for determining whether a given image is real or fake. It does so by being trained on both fake and real images of certain objects.

The second neural network, named Generator, is a decoder type of neural network which generates fake images of a given

class. Its goal is to generate samples which are indistinguishable from the real images. The adversarial training results from the fact that both networks are trained together at the same time, each learning from the other. The generator aims to fool the discriminator by generating images as similar to real ones as possible, while the discriminator is responsible for deciding whether an image is real or fake. When one network becomes better, then the least proficient one aims to improve itself.

In the following, we present the most used GANs in the generative processes.

1) Classic GAN: The initial GAN had a simple artificial neural network architecture formed by fully-connected layers, also known as multilayer perceptions [4]. Godfellow et al. proved in their paper the mathematical background required to motivate the potential and correctness of GANs. The main loss scheme employed in training the classic GAN was defined as a two-player min-max game in which the generator's goal was to minimize the loss and the discriminator's goal was to maximize its probability of assigning the correct label to both real data and fake data.

2) Conditional GAN: The Conditional Generative Adversarial Networks were introduced by Mirza and Osindero in [10] to add some constraints to the generation results. The original GAN can be extended to a Conditional GAN by simply conditioning both the discriminator and the generator on some extra information. Usually, before the input is passed forward to the generator and discriminator, some additional information is concatenated at the end of the data.

3) Deep Convolutional Generative Adversarial Networks: Modern image classifiers use convolutional neural networks to extract features from images and assign them to a pre-defined set of classes. These classifiers can be used in the context of GANs for the discriminator. The generator can also feature convolutions for the encoding part and upsampling convolutions for the decoding part. This is how Deep Convolutional GANs (DCGANs) [13] work.

B. Inpainting

The image inpainting task is defined as the task of filling in missing pixels in images. Basic inpainting methods include image processing techniques that infer the missing pixels by considering the closest existing pixels. One such example is the method proposed in [21] which uses a fast marching method [18] that expands known information to unknown regions. This method provides fast and good results for small patches. However, for large patches, due to the fact that the approach is based on local information, the quality of the results is affected.

In the deep learning context, one of the first methods that solve this task uses an encoder-decoder architecture [12] to capture local context and use it to paint the missing areas. Yeh et al. [22] attempt to use GANs to solve this problem, as they have shown impressive results on the image generation task [7]. Other works also use GANs to inpaint images. Nazeri et al. [11] tackled this task by introducing **EdgeConnect**, a two-step generative process inspired by the artistic process of painters. The authors noticed that human artists usually start by sketching, a step in which the contours are drawn, then proceeded to add details and color the illustration. The first generator network is tasked with inpainting the contours based on the grayscale input image. Next, the second generator learns to add the remaining shapes and colors based on the generated sketch and the available part of the original image.

C. Realism Enhancement

Realism enhancement is not an established task in the literature, rather it is a particular instance of the domain change class of problems. In this task, a model is trained to transform an input image from one domain (source domain) to another domain (target domain). This problem is also known as image-to-image translation or image style transfer.

The realism enhancement process in the synthetic data generation domain tries to bring a quality improvement over the traffic sign synthetic prototypes inserted in the input image. Its goal is to fit the embedded road traffic signs as naturally as possible with the semantics of the background. The alternative way is to insert the embedded prototypes without any other processing steps. However, this leads to low-quality synthetic images and weak traffic sign object detectors.

Some methods rely on Neural Style Transfer [3] that uses a feature extractor (usually a pre-trained CNN network like VGG-16, VGG-19 [19]) which extracts some hidden features from the intermediate layers of the CNN architecture. The obtained features are used in the computation of two kinds of losses: perceptual loss (content loss) and style loss. The perceptual loss encourages the creation of images that are similar to the real ones regarding the high-level content like objects, shapes etc. The style loss is an error function that measures the similarity and the correlation between the extracted features of the target image and of the initial image. The goal of the style loss is to learn the style and the texture of the initial image and to apply them to the target image.

Other methods use GANs to bring some realism to the prototype traffic signs. Here the generator is driven by the discriminator in order to capture the real distribution of the data that contains real-life road icons. There are many ways to consider the type of the generator architecture, but the most suitable is the encoding-decoding architecture. This is a type of GAN that learns the most representative features from the input images, and, based on the latent space built around them, it is able to reconstruct a new image with a more realistic icon sign.

In [8], the authors present a method that combines both the GAN work and the Neural Style Transfer (NST) component. In this way, the NST losses are turning the encoding-decoding GAN into a conditional one. The perceptual and style components are acting like some indicators during the training process of the generator.

D. Object Detection

Object detection refers to the task of identifying and localizing objects within digital images or videos [23]. It is

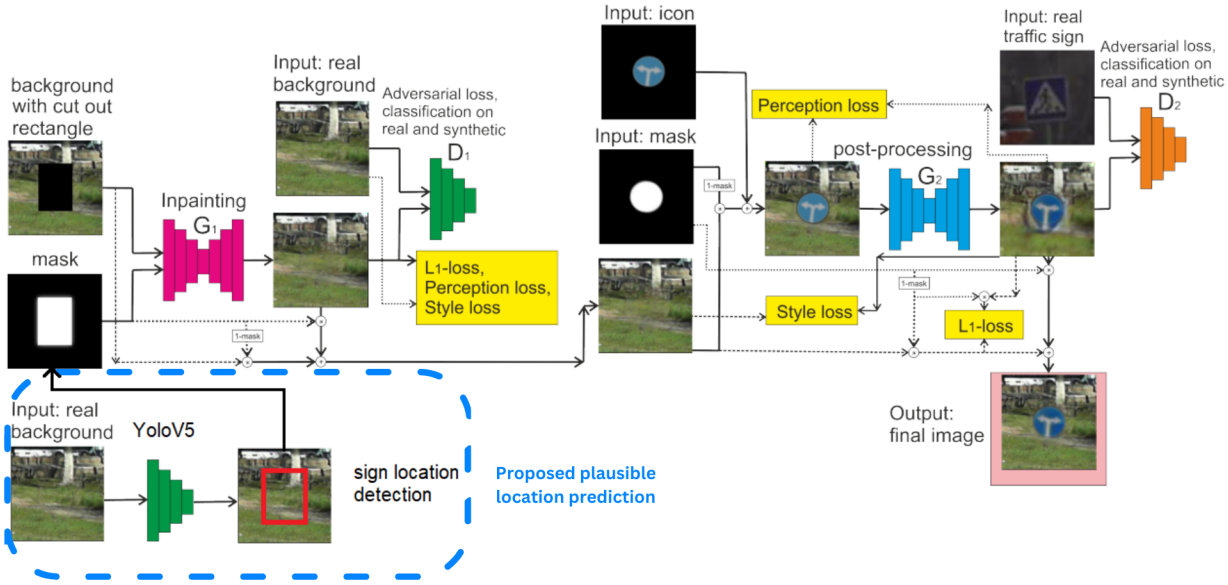


Fig. 1. Overview of the proposed approach which extends the *pasted* approach from [8]. Besides the additional location module, the architectures of the GANs are also modified.

a fundamental computer vision technique that has numerous applications, including surveillance, image retrieval, and autonomous driving. Recent advancements in deep learning and in computation power allow object detection to be performed in real-time. One notable architecture which can be used in such applications is You Only Look Once (YOLO) [14].

YOLO is a well-established single-stage detector that reformulated the object detection task into a regression problem. Initial candidate detection proposals based on template shapes, named anchors, are fine-tuned by displacing them on both the *x-axis* and *y-axis*. The model learns to perform this task through three main loss functions, which supervise the model in terms of three aspects: bounding-box shape, bounding-box location, and in multi-class problems, classification performance.

III. Approach

We propose GANERaTE, a method consisting of three steps, illustrated in Fig. 1. First, an area from the input image is chosen for the synthetic sign based on either the global distribution of traffic sign locations from our dataset or an object detector which learns to predict plausible locations. Next, we rely on a GAN for inpainting, removing undesirable details from the chosen location and filling the missing pixels with new information. Finally, we use another GAN for seamlessly integrating a synthetic sign into the inpainted region. This final step is referred to as the realism enhancement step.

Konushin et al. [8] proposed three researched methods which can generate good synthetic traffic signs that help the work of traffic sign object detectors, named *pasted*, *cycled*, and *style*. These methods are split into two categories with respect to the way the initial road traffic prototypes are placed in the image: embedded over existing real traffic signs or embedded in new locations. The *pasted* and *cycled* methods place the new icons over the existing ones, while the *style* method inserts icons in new locations. The main difference between the first two is in how the inpainting method is trained. In the *pasted* approach this component is trained to fill missing pixels from background samples which contain no traffic signs, while the *cycled* method is specifically trained to remove existing traffic signs. Our approach is inspired by the *pasted* and *style* approach as we follow the pipeline of the former one while placing the signs at new locations, as proposed in the latter.

After a plausible location is selected, as described in section III-D, the pipeline continues with the inpainting process of the patches where we want to insert some prototype traffic signs. So, the first neural network receives the patch together with the mask of the zone that we want to remove. The inputs are passed forward to the first neural network that tries to redraw the removed part according to the background. Once it was done, we insert just the redrawn crop from the patch in the initial image and the result is given as input to the second neural network.

The realism enhancement neural network tries to fit as good as possible the upcoming traffic sign that is going to be inserted in the inpainted patch. The network takes as input the inpainted patch got from the first step, a traffic sign and its mask. All of these inputs suffice in order to integrate the sign in the middle of the patch that is going to be fed into the generator of the second network. The outcome of the generator is the original patch in which the inserted sign looks much more realistic. The pipeline ends with a post processing step where just the crop with the sign from the generated patch is taken and inserted over the initial inpainted patch got from the inpainting network.

In the *pasted* approach, synthetic signs are added over

existing real signs. In order to exploit all objects from the dataset, while making the most out of generated synthetic signs we employ a model for sign location prediction, as proposed in the *style* pipeline. This allows us to place new signs in the image while keeping the real ones. We experiment with a different approach, replacing the original Variational Auto Encoder (VAE) method with a lightweight object detector trained to find plausible locations.

A. Datasets

There is a lot of research in traffic sign-related tasks, therefore there are many object detection datasets in this domain. The Mapillary Vistas [2] is one of the largest datasets from this domain with over 300.000 instances of signs from 313 different classes in 100.000 images. The images are taken world-wide with various high resolutions, providing images in a large variety of conditions such as occluded, interior, night, snowy, and others. This dataset is used in our work for most of the tasks.

Another widely-used dataset for traffic sign detection is the German Traffic Sign Detection Benchmark (GTSDB) [6]. This older dataset provides only 900 high-resolution images with 43 different types of signs. We have used this dataset for the training of both main components of our pipeline: the inpainting GAN and the realism enhancement step.

B. Inpainting Process

The first part of the pipeline implies extracting patches in the original image, removing the area from the patch where the sign will be added, and inpainting that area. If the synthetic sign is not added over an existing sign, this step is not mandatory. If the synthetic sign is added over an existing sign, the inpainting step is required to remove the old sign, as the old sign might have a different shape than the synthetic one. For example, consider the different shapes of the stop and yield signs. Once the patches were extracted and the inpainting areas are marked, we tried three options to inpaint the missing pixels.

In the first method, we have considered a simple image processing method that uses the Fast Marching method [21] for filling missing pixels starting from the known boundaries of the region to be inpainted. We have denoted this method as *Telea* in our experiments and use the implementation provided in the *open-cv2* library [1].

The second method uses a GAN to inpaint the missing pixels. Similar to the *pasted* method from [8], we have trained the GAN on patches with natural background such as trees, sky, and road. We have used 600 such patches from which we have extracted five types of crops of random rectangular shapes to be inpainted from GTSDB [6]. The rectangular shapes were randomly selected in the middle of the patch and had an aspect ratio close to a square. By randomly assigning crops to inpaint, the GAN became invariant to the size of the crop. As for the GAN details, we have implemented a discriminator with three convolution layers and a U-Net generator [16].

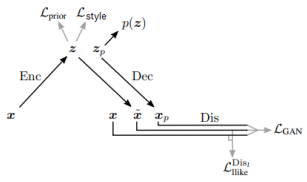
Thirdly, we have also experimented with EdgeConnect [11] to inpaint images. The method first trained a GAN to obtain the edge map and then used it as a helping representation in the second GAN that filled the missing pixels in. We have implemented EdgeConnect from scratch.

C. Realism Enhancement

The second part of the pipeline implies improving the visual quality of the prototype icon sign that was embedded into the traffic image.

In order to generate samples both diverse and realistic, we have approached the problem in the early stages with a simple GAN. The generator followed a U-Net architecture that received patches of size 128×128 pixels and compressed them into sizes of 1×1 pixels using 7 downsampled CNN blocks. In this way, the model captured just the most important features from the original image. Based on the built latent space, the generator tried to resize back the downsampled image to its original sizes through 7 upsample transposed CNN blocks. The new image should have a better fit of the embedded icon sign into the background. We have used 600 images from GTSDB Benchmark dataset [6] in order to train the model. Also, during the training process, the content and the style losses from the Neural Style Transfer [3] were applied in order to increase the convergence of semantic embedding into the background of the sign. The experimental setup considered above was chosen based on its capability to generate quality samples.

Due to the limitations imposed by the generator to reproduce new and better samples from the latent space, we have opted to change the architecture of the GAN generator into a Variational Auto Encoder GAN [9]. The generator is composed of two parts: an encoder and a decoder. The encoder tries to map the most relevant features from the input image into a distribution and not into a fixed length array as in the previous generator model attempt. The decoder takes some samples from the latent space distribution and builds a new image that should target our reconstruction goals. Fig. 2 captures the flow of the new realism enhancement neural network.



Algorithm 1 Training the VAE/GAN model

$\boldsymbol{\theta}_{\text{Enc}}, \boldsymbol{\theta}_{\text{Dec}}, \boldsymbol{\theta}_{\text{Dis}} \leftarrow$ initialize network parameters
repeat
  $\boldsymbol{X} \leftarrow$ random mini-batch from dataset
  $\boldsymbol{Z} \leftarrow \text{Enc}(\boldsymbol{X})$
  $\mathcal{L}_{\text{prior}} \leftarrow D_{\text{KL}}(q(\boldsymbol{Z}|\boldsymbol{X})\|p(\boldsymbol{Z}))$
  $\tilde{\boldsymbol{X}} \leftarrow \text{Dec}(\boldsymbol{Z})$
  $\mathcal{L}_{\text{llike}}^{\text{Dis}_l} \leftarrow -\mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{X})}[p(\text{Dis}_l(\boldsymbol{X})|\boldsymbol{Z})]$
  $\boldsymbol{Z}_p \leftarrow$ samples from prior $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
  $\boldsymbol{X}_p \leftarrow \text{Dec}(\boldsymbol{Z}_p)$
  $\mathcal{L}_{\text{GAN}} \leftarrow \log(\text{Dis}(\boldsymbol{X})) + \log(1 - \text{Dis}(\tilde{\boldsymbol{X}})) + \log(1 - \text{Dis}(\boldsymbol{X}_p))$
  // Update parameters according to gradients
  $\boldsymbol{\theta}_{\text{Enc}} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Enc}}}(\mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{llike}}^{\text{Dis}_l})$
  $\boldsymbol{\theta}_{\text{Dec}} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Dec}}}(\gamma\mathcal{L}_{\text{llike}}^{\text{Dis}_l} - \mathcal{L}_{\text{GAN}})$
  $\boldsymbol{\theta}_{\text{Dis}} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Dis}}}\mathcal{L}_{\text{GAN}}$
until deadline

Fig. 2. The overview architecture of the VAEGAN and its general way of working. Adapted from [9].

What is particular to this architecture is that the reconstruction loss of the decoder component is based on a similarity metric, and not on an element-wise reconstruction loss. The similarity metric is given by the GAN discriminator that has

the main property of learning a rich similarity measure for images, as to discriminate from "non-images".

If we denote with $Dis_l(x)$ the hidden representation of the l-th layer of the discriminator for the images x, the similarity metric is given by a Gaussian observation model for $Dis_l(x)$ with the mean $Dis(\hat{x})$ and identity covariance. Here, x is represented by the input images (the real ones), while $\hat{x}$ are the reconstructed images given by the decoder component. In this context, the reconstruction loss is computed as the probability logarithm of how likely is as the input images to be situated close to the mean of the Gaussian observation model. If the probability converges to zero, then the loss converges to $\inf$, which means that the input images are very unlikely to be sampled, according to the distribution. If this is the case, one must take into account that the distribution is computed according to the features of images given by the discriminator. Thus, the problem is not caused by the discriminator, because the feature extractor is applied both on the decoded images and the input images. We have to improve the decoder and encoder, in a such way that the features between these two domains are as close as possible. In order to do this, we aim to regularize the encoder and decoder to transfer the features of input images given by the discriminator at that stage, to the decoded images. Therefore, there is no reason to propagate the reconstruction loss to the discriminator since it causes the network to interpret the fake features as real ones or the other way around. In other words, it breaks the similarity metric.

An important note is the way how each of the three components changes their weights during the training. According to Fig. 2, the decoder is changing its weights with respect to the reconstruction and GAN losses. Moreover, in the encoder component, the reconstruction loss is applied, but it is augmented with a prior loss as well. This prior loss is used in order to transform the distribution of the latent space closer to a normal distribution. The discriminator varies just with respect to the GAN loss.

Apart from the losses from [9], we have added a style loss over the encoder and decoder components according to [3]. To emphasize the application of the style loss over the decoded images, we have multiplied the stylization losses with some weights.

We have implemented the VAEGAN architecture from scratch and validated it on the GTSDB [6] dataset. Only one sign was considered: the prohibitory sign. The training set consists of 600 images of size 128×128 during training, and the number of training epochs has an upper bound equal to 250 epochs. This kind of setup was chosen by putting into balance the performance of the generator and the computational time.

D. Plausible Location

Creating realistic synthetic traffic signs and integrating them seamlessly into the original picture are important parts of our pipeline. The traffic signs should be embedded in plausible locations in order to maximize the realism of the resulting pictures.

We have used two approaches for this step. The first approach is image-agnostic and is based on data analysis performed on the whole dataset. A heatmap based on all real traffic sign locations from the dataset was computed. Based on the found distribution, locations are generated, without taking into consideration the input image. The second approach was developed to generate specific locations for each image. We have achieved this by training the Yolo object detector on 1200 samples from the Mapillary dataset in which all traffic signs were inpainted using the Telea method. Through this process, we hoped the model would learn to associate elements such as poles with traffic signs and would be able to predict realistic locations in unseen images.

IV. Results

A. Evaluation Metrics

The literature contains various metrics for evaluating image generation models. One of the metrics used in order to provide a quantitative evaluation is the *Inception Score* (IS) [17]. It is computed by using the class probabilities of a pre-trained InceptionV3 network [20] on the generated images. The higher quality generated images should lead to confident predictions, while lower quality images confuse the pre-trained model.

Although useful, the Inception Score it is limited in the sense that it does not compare the generated images with the real distribution of data. As a response, the *Fréchet Inception Distance* (FID) was proposed [5]. Similarly to IS, FID uses the pre-trained InceptionV3, but the metric is computed by using both the generated and the real images. Using the features obtained by the convolutional part of the deep neural network, the difference between real and generated is computed. Nowadays, the Fréchet Inception Distance has replaced IS in almost all applications.

We evaluated the performance of an object detector on the validation set consisting of real images after it has been trained on a mix of real and synthetically augmented images. We report values for three main metrics in these experiments: Precision, Recall and Mean Average Precision **(mAP)**. The first two are widely-used metrics in multiple tasks. Mean-Average Precision is used in information retrieval and object detection. It measures the performance of a system by calculating the average precision across multiple levels of precision. In the context of object detection, mAP considers the precision and recall of the detected objects. This metric is computed at various thresholds, for example, mAP@50 is a specific variation of mAP in which a detection candidate is considered correct if it has at least 50% overlap with a real object bounding box. We measured our performance using mAP@50 and a version in which we averaged the mAP value at different thresholds, more specifically all values from 50 to 95 with a step of 5. We denote this by mAP@50:95:5.

B. Inpainting Evaluation

In terms of metrics, the GAN-based approach and Telea inpainting have achieved a similar performance, as it can be

seen in Table I. Both outperformed the EdgeConnect method by notable margins.

Method	Class	IS	FID
EdgeConnect	stop	1.6	206
GAN		**2.06**	126
Telea		**2.06**	**123**
EdgeConnect	yield	1.6	186
GAN		**2.27**	**119**
Telea		2.26	120

TABLE I
GAN METRICS' VALUES FOR INPAINTING METHODS.

The visual results of the three investigated inpainting approaches are displayed in Fig. 3. The same conclusion can be drawn from the images and the GAN metrics. EdgeConnect leads to the worst result. This issue could be the result of our approach, as the simple GAN is trained on patches, making it more suited for this application. EdgeConnect however, is trained on full-resolution images. When applied in the pipeline, the initial inpainting obtained by EdgeConnect is just a small part of the resulting output, meaning that the extracted inpainted crop has to be upscaled, therefore diminishing the level of detail considerably.

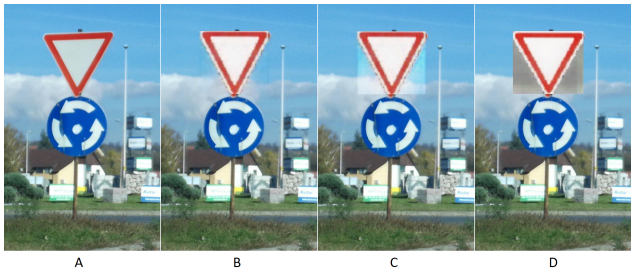


Fig. 3. Visual comparison of inpainting results: (A) original image; (B) Telea; (C) Generator from GAN; (D) EdgeConnect.

C. Realistic Enhancement Evaluation

The results of the realism enhancement network are displayed in Fig. 4. This approach follows the details of the *pasted* method from [8]. We enhance the generated sign using Neural Style Transfer [3] to propagate the style of the background on the sign.

Fig. 4. Realism enhancement results by applying pasted method: (A) input image; (B) generator outcome; (C) Neural Style Transfer applied on generator outcome; (D) post-processing step on the generated image.

Due to the poor artificial results, we continue with the experiments involving Variational Autoencoders [9]. The results of the VAEGAN architecture from the testing stage can be observed in Fig. 5. The quality of the results is satisfactory. The reconstruction of the sign is preserved and features such as shine, brightness, shadow, texture, are all adapted with respect to the background of the initial images. While the background of the generated images is not learnt, we do not have to since it is replaced with the background of the input image.

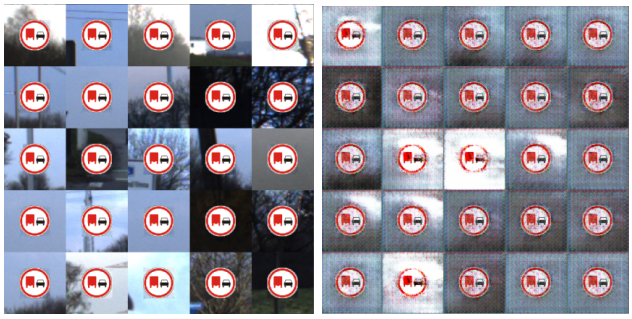


Fig. 5. Realism enhancement results. Left: input images; Right: raw results.

In Fig. 6, the capability of the generator is showcased to generate samples from the training data. The top-row images are generated by the decoder component, with their corresponding input images on the second row. The last row contains the L2 distances (Euclidean distance) between the first two rows. Notice the reconstruction of the traffic sign is qualitative, and the style is applied with respect to the background of the image. However, the texture and style on the target images are only sightly applied. We believe this is caused by the discriminator. During the training, the discriminator learns how to discriminate between the real and generated images, which leads the discriminator to capture a small amount of features regarding the stylization to be able to distinguish between the two data domains. Thus, the similarity metric is poor due to the unbalance between the generator performance and the discriminator performance.

D. Evaluation Pipeline

The purpose of synthetic traffic signs is to improve the performance of object detectors on specific classes. Therefore, we evaluated how much does the performance of the YOLOv5 object detector improve when adding synthetic signs.

The dataset we experimented on is the Mapillary dataset [2]. We filtered the images with the stop and yield traffic signs, obtaining 295 and 399 images with 347 and 500 real sign instances respectively. If there were different signs in an image, we considered only the ones of interest. The images were randomly split into 80:20 training and validation subsets. The validation subsets were comprised of original images without any synthetic signs.

We began experimenting by adding only 20 synthetic images to see whether such a small number of synthetic objects improves performance. We trained the model for 30 epochs with three random seeds and reported the mean and standard deviation in Table IV for the *yield* sign, and Table II for the *stop* sign. We included the precision, recall, mAP50 and mAP0.5:0.95 scores for each experiment. *Gen* denotes the Generator from the GAN, *Edge* denotes the EdgeConnect

Fig. 6. Decoder results for the VAEGAN architecture in which the style loss was applied on the both encoder and decoder.

Aug	Precision	Recall	mAP50	mAP0.5:0.95
No	0.795±0.139	0.720±0.105	0.743±0.126	0.514±0.081
Add20 Gen	**0.897±0.124**	0.712±0.139	0.757±0.121	0.534±0.081
Add20 Gen-Det	0.860±0.064	**0.729±0.110**	0.752±0.103	0.533±0.065
Add20 Edge	0.864±0.097	0.717±0.105	0.750±0.092	0.532±0.072
Add20 Telea	0.842±0.099	0.710±0.140	0.749±0.118	**0.536±0.085**
Add20 Telea-Det	0.852±0.113	0.723±0.120	**0.759±0.086**	0.525±0.083

TABLE II
OBJECT DETECTION-BASED EVALUATION ON **STOP** SIGN USING DIFFERENT INPAINTING METHODS.

Aug	Precision	Recall	mAP50	mAP0.5:0.95
No	0.908±0.080	0.548±0.059	0.624±0.082	0.388±0.061
Add20 Gen	0.897±0.068	0.545±0.040	0.649±0.083	0.395±0.063
Add20 Gen-Det	0.866±0.066	0.584±0.085	0.650±0.075	0.384±0.066
Add20 Edge	0.945±0.029	0.545±0.040	0.618±0.083	0.388±0.067
Add20 Telea	**0.947±0.029**	0.547±0.102	0.636±0.069	0.389±0.055
Add20 Telea-Det	0.927±0.108	**0.587±0.023**	**0.664±0.066**	**0.406±0.079**

TABLE IV
OBJECT DETECTION-BASED EVALUATION ON **YIELD** SIGN USING DIFFERENT INPAINTING METHODS.

method, and *Telea* means that the image processing technique was used. *Det* results were obtained on signs with locations predicted by an object detector. The results show an increase in performance when using synthetic sign augmentations. The best results were achieved using the simple Telea inpainting method with the pre-trained object detector for plausible locations.

Aug	Precision	Recall	mAP50	mAP0.5:0.95
No	0.908±0.080	0.548±0.059	0.624±0.082	0.388±0.061
Add20 Gen	0.897±0.068	0.545±0.040	0.649±0.083	0.395±0.063
Add20 Telea	**0.947±0.029**	0.547±0.102	0.636±0.069	0.389±0.055
Add80 Gen	0.897±0.043	**0.607±0.102**	0.668±0.081	**0.424±0.82**
Add80 Telea	0.913±0.067	0.606±0.087	0.671±0.076	0.399±0.060
Add400 Gen	0.880±0.007	0.604±0.102	**0.685±0.097**	0.419±0.084
Add400 Telea	0.911±0.117	0.578±0.088	0.660±0.063	0.400±0.066

TABLE III
OBJECT DETECTION-BASED EVALUATION WITH A VARIABLE AMOUNT OF SYNTHETIC SIGNS FOR **YIELD** TRAFFIC SING.

We then compared the inpainting methods and observed that Generator and Telea led to a better performance than Edge-Connect. One possible reason for this is the way EdgeConnect was implemented. The network was trained to work with the entire image, not with specific patches. This caused the inpainting of small patches to be less realistic, as the image is first downsized and the patch to be inpainted is also downsized, leading to a high loss of details. After the downsized image is given as input to the model, it outputs an image of the same size which must be extrapolated back to the original size, losing more details in the process.

We also added more synthetic signs to investigate whether a more significant increase in performance is achievable. Specifically, we added 80 synthetic signs in 40 images (2 signs per image) and 400 synthetic signs in 404 images (10 signs per image). The number of training epochs was increased from 30 to 40 since there are more signs. Again, three seeds were performed for each experiment, and we used the pre-trained object detector for plausible sign locations to place the synthetic signs.

Aug	Precision	Recall	mAP50	mAP0.5:0.95
No	0.795±0.139	0.720±0.105	0.743±0.126	0.514±0.081
Add20 Gen	0.897±0.124	0.712±0.139	**0.757±0.121**	0.534±0.081
Add20 Telea	0.842±0.099	0.710±0.140	0.749±0.118	0.536±0.085
Add80 Gen	0.779±0.142	0.690±0.136	0.735±0.095	**0.538±0.076**
Add80 Telea	0.835±0.113	0.698±0.100	0.743±0.092	0.528±0.083
Add400 Gen	0.770±0.140	**0.727±0.063**	0.737±0.092	0.523±0.082
Add400 Telea	**0.917±0.085**	0.656±0.110	0.741±0.082	0.535±0.069

TABLE V
OBJECT DETECTION-BASED EVALUATION WITH A VARIABLE AMOUNT OF SYNTHETIC SIGNS FOR **STOP** TRAFFIC SING.

For the *yield* class, we observe in Table III that more signs do indeed lead to better mAP scores by up to 6.1%. Moreover, the results are statistically comparable since the standard deviations do not overlap. However, on the *stop* class, the performance with more signs in the images does not improve as much, but it is still considerable. Note that the best performance in these experiments was obtained when using the inpainting generator instead of the Telea method. The results for this class are displayed in Table V.

Comparing to [8], they report the Area Uder the Curve (AUC) score on trained object detectors. The largest increase they reach is of 6.4% when training on both real and synthetic rare signs. We reach a similar performance when adding many synthetic signs and inpainting with the generator III.

The answer to our first research question is that augmenting images with synthetic signs can improve the performance of the object detection task. The answer to the second research is not so clear since different numbers of extra synthetic signs perform differently depending on the considered metric. Adding 400 extra images leads to good performance in terms of precision and recall. On the other hand, the proficiency of the object detector measured in terms of the mAP is maximized by having only 20 to 80 additional synthetic signs.

V. Conclusions and Future Considerations

In this paper, we showed that enhancing datasets with realistic traffic signs can lead to object detector performance improvements. Although the best approach is empowered by deep learning methods, we can see that classic computer vision approaches like the Telea inpainting algorithm help us achieve competitive results at a fraction of the computational cost.

Realistic sign enhancement has brought satisfying results by using the VAEGAN architecture. This kind of model relies on the similarity metric of the features being extracted in the process of generating new realistic samples. The content and style of the samples are applied well, bringing a realism touch over the synthetic icon signs. However, the process degenerates quickly due to the proficient performance of the discriminator.

Some aspects that can be improved in the future are:

- Compare our approach to the baseline method from [8] using the same metrics as them;
- Further experiments should be conducted with EdgeConnect in order to fully exploit this complex architecture.

References

[1] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[2] Christian Ertler et al. "The mapillary traffic sign dataset for detection and classification on a global scale". In: *Computer Vision–ECCV 2020 Proceedings, Part XXIII 16*. Springer. 2020, pp. 68–84.

[3] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "A neural algorithm of artistic style". In: *preprint arXiv:1508.06576* (2015), pp. 1–16.

[4] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[5] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in neural information processing systems* 30 (2017), pp. 6629–6640.

[6] Sebastian Houben et al. "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark". In: *The 2013 international joint conference on neural networks (IJCNN)*. IEEE. 2013, pp. 1–8.

[7] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *CVPR*. 2019, pp. 4401–4410.

[8] Anton Konushin, Boris Faizov, and Vlad Shakhuro. "Road images augmentation with synthetic traffic signs using neural networks". In: *preprint arXiv:2101.04927* (2021), pp. 1–15.

[9] Anders Boesen Lindbo Larsen et al. "Autoencoding beyond pixels using a learned similarity metric". In: *International conference on machine learning*. PMLR. 2016, pp. 1558–1566.

[10] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *preprint arXiv:1411.1784* (2014), pp. 1–7.

[11] Kamyar Nazeri et al. "EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning". In: *preprint arXiv:1901.00212* (2019), pp. 1–17.

[12] Deepak Pathak et al. "Context Encoders: Feature Learning by Inpainting". In: *CVPR*. 2016, pp. 2536–2544.

[13] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *preprint arXiv:1511.06434* (2015), pp. 1–16.

[14] Joseph Redmon and Ali Farhadi. "Yolov3: An Incremental Improvement". In: *preprint arXiv:1804.02767* (2018), pp. 1–6.

[15] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *CVPR*. 2016, pp. 779–788.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention: 18th International Conference Proceedings, Part III*. 2015, pp. 234–241.

[17] Tim Salimans et al. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems* 29 (2016), pp. 2226–2234.

[18] James A Sethian. "A fast marching level set method for monotonically advancing fronts." In: *Proceedings of the National Academy of Sciences* 93.4 (1996), pp. 1591–1595.

[19] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *preprint arXiv:1409.1556* (2014), pp. 1–14.

[20] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *CVPR*. 2016, pp. 2818–2826.

[21] Alexandru Telea. "An Image Inpainting Technique Based on the Fast Marching Method". In: *Journal of graphics tools* 9.1 (2004), pp. 23–34.

[22] Raymond A Yeh et al. "Semantic image inpainting with deep generative models". In: *CVPR*. 2017, pp. 5485–5493.

[23] Syed Sahil Abbas Zaidi et al. "A survey of modern deep learning based object detection models". In: *Digital Signal Processing* (2022), p. 103514.