Realistic Text Replacement With Non-Uniform Style Conditioning

ARSENY NERINOVSKY, IGOR BUZHINSKY, AND ANDREY FILCHENKOV (Member, IEEE)

Computer Technologies Laboratory, ITMO University, 197101 St. Petersburg, Russia
Department of Electrical Engineering and Automation, Aalto University, 02150 Espoo, Finland
Corresponding author: Arseny Nerinovsky (arseny-n@yandex.ru)

This work was supported by the Government of the Russian Federation under Grant 08-08.

ABSTRACT In this work, we study the possibility of realistic text replacement. The goal of realistic text replacement is to replace text present in the image with user-supplied text. The replacement should be performed in a way that will not allow distinguishing the resulting image from the original one. We achieve this goal by developing a novel non-uniform style conditioning layer and apply it to an encoder-decoder ResNet based architecture. The resulting model is a single-stage model, with no post-processing. We train the model with a combination of adversarial, style, content and $L_1$ losses. Qualitative and quantitative evaluations show that the model achieves realistic text replacement and outperforms existing approaches in multilingual and challenging scenarios. Quantitative evaluation is performed with direct metrics, like SSIM and PSNR, and proxy metrics based on the performance of a text recognition model. The proposed model has several potential applications in augmented reality.

INDEX TERMS GAN, style conditioning, text replacement.

I. INTRODUCTION

The task of realistic text replacement [1] could be formulated as follows: replace text present in an image with arbitrary user-supplied text in a way that will not allow distinguishing the resulting image from the original one. One of the technologies which stand to primarily benefit from realistic text replacement is augmented reality (AR) text translation. It allows replacing text in a foreign language with text in the native language of the user. An even more promising technology is the coupling of text replacement and text translation with smart glasses devices, where the translation and substitution of text could happen in real time thus allowing the user to seamlessly navigate foreign language environments.

Realistic text replacement is quite challenging since text is usually present in a variety of styles and on a variety of backgrounds. An illustration of the text replacement task could be seen in Figure 1, where all selected text (denoted by pink polygons) in the source image is substituted with the string “hello.”

We solve the task of realistic text replacement with a generative adversarial network (GAN) [2] based on paper [3]. The generator is based on a ResNet encoder-decoder architecture [4]. Unlike in previous works, text replacement is made with one forward pass through the network without post-processing. In order to perform text replacement, two images are required—a content image, which parametrizes what is inpainted, and a style image, which parametrizes the style of the inpainted images. Also masks denoting regions of the image where text is present are required. The parametrization of the inpainted areas is performed by replacing areas of
FIGURE 2. Scheme of the proposed architecture.

The challenge of different text styles was addressed by the introduction of a novel non-uniform conditioning layer called PatchAdaIn. PatchAdaIn allows us to extract user-delimited areas of style information from the style image and apply it to areas of the content image. This allowed us to stylize different text instances present in the same image with different styles.

The major contributions of this paper are:

- PatchAdaIn, a non-uniform conditional normalization layer which allows applying different styles to different image parts;
- a network achieving realistic text replacement, which we call Patch-Style GAN (PsGAN).

This paper is structured as follows. In Section II, we describe works related to the proposed approach. Then, in Section III, we describe the proposed approach: the non-uniform conditioning layer in Section III-A and the adopted architecture in Section III-B. Then we describe the performed experiments in Section IV. Section V concludes the paper.

II. RELATED WORK

A. GENERATIVE ADVERSARIAL NETWORKS

GANs [2] are a machine learning framework consisting of typically two networks: a generator network and a discriminator network. The task of the generator is to create samples which best resemble samples from the training dataset. The discriminator is tasked with differentiating generated and real samples. Both networks are trained together, but with different loss functions.

In the original GAN formulation, the generator network maps samples from a normal distribution to images. The work of [5] introduces conditional GANs (cGANs). cGANs allow controlling their output by supplying a class label. Several improvements to the cGANs model are proposed in works [6], [7]. They mainly differ in how the class label is supplied to the network. Several works explored the conditioning of GANs not only bound to a single class label. Variants of conditioning by text [8], bounding boxes and key-points [9], or images [3] exist. The latter is the most relevant to the present work.

To allow image conditioning, the authors of [3] propose to use a Unet [10] network for the generator, and the proposed model is called Pix2Pix. The network maps a source image to a target image. Examples of source-target image pairs are segmentation maps and images from which such segmentation maps originate, black and white images and color variants of the former, and images with absent regions and the original images. The authors successfully apply the Pix2Pix network to a variety of source-target pairs. They refer to Pix2Pix as to a model for paired image translation due to its generality.

B. CONDITIONING BY MODULATION

Batch normalization (BN) is an important part of many state-of-the-art neural network architectures. For example, the authors of DCGAN [11], which is the first GAN that successfully applied a convolutional encoder and decoder, partly ascribe their success to aggressive use of BN. BN is composed of a normalization step followed by a modulation step. During the normalization step, the input is normalized by subtracting the mean $\mu_{b,h,w}$ and then dividing by the standard deviation $\sigma_{b,h,w}$:

$$\hat{y}(x) = \frac{x - \mu_{b,h,w}(x)}{\sigma_{b,h,w}(x)},$$

where the mean and standard deviation are computed across the batch, height and width dimensions. The modulation step is defined as

$$BN(x) = \gamma \hat{y}(x) + \beta,$$

where $\gamma$ and $\beta$ are learned scale and shift parameters.

Several variants of BN have been developed [12], [13]. The variation of BN most relevant to this work is instance normalization. First introduced in [14], instance normalization was developed for aiding style transfer tasks. The definition of instance normalization differs from BN only in the way the statistics are computed. Unlike BN, instance normalization
computes statistics across the channel dimension:
\[
\text{In}(x) = \gamma \frac{x - \mu_{h,w}(x)}{\sigma_{h,w}(x)} + \beta,
\]
where \(\mu_{h,w}\) and \(\sigma_{h,w}\) are the mean and standard deviation computed across the channel dimension.

Conditional instance normalization is an extension of instance normalization proposed in [15]. The authors of [15] substitute the modulation parameters \(\gamma\) and \(\beta\) by \(\gamma'\) and \(\beta'\) which are parametrized by the style image \(s\). This approach is further improved by AdaIn [16], where instead of the learned parameters feature statistics are used. The latter allows avoiding pre-learned parameters. The AdaIn layer is formulated as
\[
\text{AdaIn}(x, y) = \sigma_{h,w}(y) \frac{x - \mu_{h,w}(x)}{\sigma_{h,w}(x)} + \mu_{h,w}(y).
\]

Conditional modulation was also later applied with different tweaks in many major GAN works, such as [17]–[19]. In general, the approach of conditional modulation of the network features has quite a varied applicability ranging from visual question answering [20] to domain adaptation [21]. Conditional modulation is formalized in [22] and given a more detailed treatment in [23].

C. REALISTIC TEXT REPLACEMENT

Several works in the area of image generation are connected with the generation of text images. The most related to this paper are the works [1] [24]. The authors of [1] [24] generate realistic text replacements. Their proposed architectures take as input a style image containing arbitrary text in a user-defined style and an input text image containing user-supplied text in a predefined style. Each image contains solely the stylized text.

Both works split the task between three modules: a text conversion module, a background inpainting module and a fusion module. In both works the modules perform very similar tasks. The text conversion module converts an image containing source text to an image where the source text has the correct color and style but no background. The background inpainting module is tasked with erasing text from the style image, leaving only the background. The task of the fusion module is to fuse the conversion module results and the background inpainting module results. The key difference is that [1] uses a text skeleton based conversion module, while [24] uses a control points based one. Also the modules of [1] are learned independently while [24] proposes a fully differentiable architecture.

Another work related to image generation and text is [25]. Its main focus is the augmentation of datasets for text recognition. The authors of [25] propose an adaptation of the Pix2Pix network to realistically color text images. The proposed architecture is composed of a combination of two Pix2Pix generators. The task of the first generator is to generate a colored version of the text area and its surroundings. The task of the second generator is to fill the remaining background. The work [26] considers the task of text inpainting in more general settings and proposes to divide geometry and appearance synthesis. This approach is applicable to realistic insertion of glasses and hats into portraits.

III. PROPOSED APPROACH

To achieve realistic text replacement, we use an extension of the paired image translation paradigm introduced in [3]. The paired image translation paradigm could be formalized as:
\[
\forall \{A_i\}_{i=0}^{N} \text{ and target } \{T_i\}_{i=0}^{N} \text{ images, find a model } F \text{ capable of mapping source images to target ones: } F(A_i) = T_i.
\]

Our proposed task formulation is a specialized extension of the paired image translation paradigm. We constrain the source image to images with text regions substituted by edge maps. We refer to the source image as to the content image since the content of the text present in the generated images is expected to be equal to the content of the text in the content image. We also introduce one more input image—the style image. The latter parameterizes the style of the inpainted regions—in particular, the inpainted text is expected to have the same background and foreground colour. We refer to the content image as to the style image. To create the content image we use an edge detection algorithm. The use of edges has the benefit of relieving us from the development of a specialized dataset.

The resulting mapping is \(F' : \forall i \ F'(A_i, S_i, M_i) = T_i\), where \(M_i = \{M_{ij}\}\) is a set of masks denoting the areas occupied by text and \(S_i\) is the style image. The style image is equal to the target image. Each mask \(M_{ij}\) is 1 where a style-uniform region of text is present and 0 everywhere else. Examples of content, style, target and mask images could be seen in Figure 2. The key notations used to describe the proposed approach are summarized in Table 1.

| Variable | Description |
|----------|-------------|
| \(A_i, T_i, S_i\) | \(i\)-th source, target and style image, respectively |
| \(M_i\) | \(i\)-th set of masks |
| \(M_{ij}\) | \(j\)-th mask corresponding to the \(i\)-th image |
| \(s, c\) | Style and content features supplied to the PatchAdaIn or PatchAdaInBg layer, respectively |

A. NON-UNIFORM CONDITIONAL NORMALIZATION

Unlike earlier works [1], [25], we design an architecture capable of text replacement of several text areas in one forward pass. To accomplish this, we introduce a novel non-uniform conditional normalization layer. Our design is based on the AdaIn layer (2). Like AdaIn, we demodulate features by subtracting the mean and dividing by the variance:
\[
c = \frac{x - \mu_{h,w}(x)}{\sigma_{h,w}(x)}.\]

We adapt the layer modulation by introducing an additional parameter—style patches areas of the image having a uniform style. Like AdaIn, we use statistics of features extracted form
a style image to modulate the demodulated layer inputs. We will refer to such features as to style features and denote them as \( s \). Such modulation is to be performed not on the whole image but on predefined areas delimited by the style patches.

In order to compute the output of the PatchAdaIn layer we iterate through all the patches and extract mean and standard deviation statistics of the areas denoted by the patches from the style features.\(^1\) Both statistics are calculated across the width and height dimension. Then we subtract the mean and divide by the standard deviation but do so only in the region denoted by the mask:

\[
    r_j = c \odot \sigma_{h,w}(M_{ij}, s) + \mu_{h,w}(M_{ij}, s),
\]

\[
    \text{PatchAdaIn}(s, c, M_I) = \sum_{j=1}^{\vert M_I \vert} r_j, \tag{3}
\]

\[
    \text{PatchAdaInBg}(s, c, M_I) = \text{PatchAdaIn}(s, c, M_I) + c \odot \sigma_{h,w}(M_{bg}, s) + \mu_{h,w}(M_{bg}, s). \tag{4}
\]

\( r_j \) \( (M_{ij}) \) \( (M_{bg}) \) \( \sigma_{h,w} \) \( \mu_{h,w} \) \( \odot \) \( \odot \)

FIGURE 3. Scheme of the proposed PatchAdaIn layer.

The PatchAdaInBg layer is computed by introducing an additional step where we modulate all the space not covered by any masks. We refer to such space as to the background. We perform modulation in a similar fashion: extract the mean and standard deviation and respectively subtract and divide the content features by them. The statistics are extracted from the style image:

\[
    M_{bg} = 1 - \sum_{j=1}^{\vert M_I \vert} M_{ij}, \tag{5}
\]

\[
    \text{PatchAdaInBg}(s, c, M_I) = \text{PatchAdaIn}(s, c, M_I) + c \odot \sigma_{h,w}(M_{bg}, s) + \mu_{h,w}(M_{bg}, s). \tag{6}
\]

\( \text{PatchAdaIn} \) \( \text{PatchAdaInBg} \) \( M_{bg} \) \( \sum \) \( (M_{ij}) \) \( (M_{bg}) \)

\(^1\)We do not directly iterate through the masks, but use PyTorch broadcasting. Although the implementation still requires a loop though the batch dimension, the resulting code has a negligible performance penalty.

### B. ARCHITECTURE

The proposed architecture consists of an encoder, several residual blocks (ResBlocks), and a decoder. The scheme could be seen in Figure 2. Conditioning is performed in the encoder and in all the residual blocks. Conditioning in the encoder was performed on features extracted from the style encoder. The style branch is identical to the encoder of the main network with one additional residual block added at the end.

In order to allow conditioning, we changed the structure of the ResBlock. The default ResBlock is structured as a pair of consecutive convolutions followed by a normalization layer. The convolutions are separated by a ReLU and a dropout layer. The modified conditioned residual blocks differed from their regular counterparts by the last normalization layer, which was substituted with a conditional PatchAdaIn normalization layer. A scheme of the proposed ResBlock could be seen in Figure 4.

The training is performed with a combination of \( L_1 \), style, content and adversarial losses as described in the Pix2Pix paper \cite{pix2pix}. Several successful examples generated by the model could be seen in Figure 5. The images were generated by first inpainting the text in the image and then substituting the text area in the original image with the inpainting result.

### IV. EXPERIMENTS

#### A. DATA

We used several text detection datasets: ICDAR MLT \cite{icdar}, COCO Text \cite{coco}, UberText \cite{ubertext} and SynthText \cite{synthtext}. All datasets contain images and polygons delimiting areas where text is present. Several samples could be seen in Figure 6.

We estimated the approximate size of the dataset based on the numbers required by sketch-to-image models and by image inpainting models. Our findings suggested that a dataset in the range of 100,000 samples will be enough, since Pix2Pix was trained as a sketch to image model on 137 thousand image pairs from the Amazon Handbag images dataset \cite{amazon}. The inpainting model from \cite{inpainting} was successfully trained on 27 thousand images from CELEBA-HQ \cite{celeba} and 50 thousand inpainting masks.

To generate data for training, we used data from text detection datasets. One peculiarity of the data is that the distribution of polygon sizes is skewed towards smaller polygons. The polygons could be so small that the text delimited by them becomes invisible. As the input to the network, we should supply a square image with reasonably sized and reasonably dense text. The nature of the data makes supplying every text instance to the network unfeasible. Thus, we rejected polygons whose largest dimension spans under 200 pixels in length. The total size of the agglomerate dataset is 245,225 samples.

#### B. METRICS

The quality of the resulting images was assessed by measuring the validation score of a pre-trained CRNN model.
from [34] for text recognition. A good validation performance is indicative of good resulting images. Two sets of metrics were estimated, namely a global and per image one. Both sets were composed of the Accuracy and WER metric. The two sets were different in the way averaging was performed. The global metrics were averaged on a per dataset basis, while the per-image metrics were averaged on a per image basis. We refer to such metrics as to validation proxy.
TABLE 2. Proxy metrics (PMs).

| Context of replacement | Data restrictions | Replacement context | Data restrictions |
|------------------------|-------------------|---------------------|------------------|
| Measures the validation accuracy of a text recognition network trained on generated images | Training PM | General replacement | None |
| Measures the validation accuracy of a text recognition network on generated images | Validation PM | General replacement | None |
| | Multilingual validation PM | Replacement in a multilingual setting | Source images restricted to a single script |
| | Varied-length validation PM | Replacement in a challenging setting | Specifically sampled target texts |

TABLE 3. Validation metrics of a pretrained CRNN network on ICDAR MLT.

| Model | Accuracy | WER | Per image accuracy | Per image WER |
|-------|----------|-----|-------------------|---------------|
| PsGAN (ours) with PatchAdaIn | 0.684 | 0.0656 | 0.697 | 0.0617 |
| PsGAN (ours) with PatchAdaInBG | 0.653 | 0.0708 | 0.668 | 0.0680 |
| Raw | 0.813 | 0.0384 | 0.824 | 0.0323 |
| Generated by [1] | 0.544 | 0.0901 | 0.517 | 0.1020 |

Another way to assess the performance of the inpainting model is to enhance the training set of a text recognition model with data generated by the inpainting model. An increase in the training score will be indicative of good inpainting model. We call this set of metrics training proxy metrics.

In order to further investigate the model performance we developed several more detailed metrics. We assessed the ability of the model to replace foreign text by calculating the validation proxy metric on images from ICDAR MLT restricted to a single script. We will refer to this metric as multilingual validation proxy metric.

During the usual validation proxy metric calculation we used to replace instances of text with text of equal length. In order to assess the ability of the model to replace text of arbitrary lengths we introduce the varied-length validation proxy metric. To calculate it we varied the length of the target text by a predefined delta, i.e., if we had an image with a source word of length 8 and the predefined delta was set to −2 then we sampled a replacement word of length 6.

Finally, apart from proxy metrics we also calculated standard image quality metrics: SSIM, PSNR [35] and $L_1$.

C. RESULTS

Our best model is currently capable of reaching a per image accuracy of 0.697, which is comparable to the accuracy of 0.824 achieved by the model on raw data. We outperform the work of [1], which achieves a per image accuracy of 0.517. The values of the validation proxy metrics could be seen in Table 3. We also outperform [1] on ICDAR13 and ICDAR15 as could be seen in Figure 8. Our best performing variant of the model is the one equipped with PatchAdaIn. All latter metrics are calculated with a PatchAdaIn conditioning layer.

Multilingual and varied proxy metrics are available in Figures 7 and 8. Image quality metrics could be found in Table 5.

In [25], proxy metrics are trained to assess the model quality. The authors of [25] train a model on 8 million synthetically generated images. In our case, because of the size of our network, such validation would be unfeasible. Instead, we augment the training dataset with additional synthetic images. The number of synthetic images is equal to 25% of the number of original training images. The model trained on the dataset augmented with synthetic images achieves a lower WER score. The training proxy metrics could be seen in Table 4.

D. DISCUSSION

As the validation proxy metrics reveal, PatchAdaIn conditioning outperforms PatchAdaInBG conditioning.
We conjecture that this happens since the background information is not used in any meaningful way and as a result conditioning by it only complicates the training process.

Also we want to note that, although the image quality metrics reveal an above average performance, upon close inspection we found a label leakage problem in the data used for validation. For the correct calculation of image quality metrics two images are needed: a reference image and an image to which a certain operation was applied. Both images should have the same contents.

In our case, the only way we could obtain a pair of images with the same contents was to apply the training data pipeline to images from the validation set. The content image is obtained by applying an edge detector to the areas of text. The presence of edge detector artifacts introduces a leak since no such artifacts would be present during inference. We still provide image quality metrics for completeness, and also note that the model was trained solely in images from the MLT dataset and data from ICDAR13 and ICDAR15 images has never been seen by the model before. We also note that during the calculation of proxy metrics we replaced the areas of text with edges of text produced by placing the edges of a randomly sampled word on a blank background.

Although our method outperforms [1], it still suffers from several common artifacts. For example, textured backgrounds pose a challenge to the method, since they are often substituted with a blurred version. Also, if the source text is smaller than the target text, the network seems to have difficulties in assigning colors to the correct areas, and as a result color spills may occur. Images with examples of inpainting artifacts could be found in Figure 9.
We were unable to compare the proposed method experimentally with works other than [1] due to the inavailability of their source code, but we note that the proposed method is simpler than its analogs [1, 24] and thus faster and easier to train. This is partly due to the absence of any font adaptation mechanism. We plan to develop such a mechanism in future works.

We also further note that the proposed approach currently only works for English as a target language but it could be extended to work on target languages other than English without any structural changes. The extension only requires a dataset of adequate size and the retraining of the model on the non-English data.

V. CONCLUSION
This paper proposes a feedforward model for realistic text replacement based on a ResBlock encoder-decoder architecture. A novel non-uniform conditioning normalization layer is introduced to allow the application of different styles to different parts of the image. The proposed models outperform previous works on all the measured metrics and achieve high image realism.

Since the developed model could be potentially used as an enhancement to translation technology, we plan to expand the scope of the model to multilingual scenarios. Another area of future work is the application of the model to mobile-compute-constrained environments.

REFERENCES
[1] L. Wu, C. Zhang, J. Liu, J. Han, J. Liu, E. Ding, and X. Bai, “Editing text in the wild,” in Proc. 27th ACM Int. Conf. Multimedia, 2019, pp. 1500–1508.
[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 2672–2680.
[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1125–1134.
[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.
[5] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” 2014, arXiv:1411.1784. [Online]. Available: https://arxiv.org/abs/1411.1784
[6] T. Miyato and M. Koyama, “CGANs with projection discriminator,” 2018, arXiv:1802.05637. [Online]. Available: https://arxiv.org/abs/1802.05637
[7] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, “Invertible conditional GANs for image editing,” 2016, arXiv:1611.06355. [Online]. Available: https://arxiv.org/abs/1611.06355
[8] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, “AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1316–1324.
[9] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, “Learning what and where to draw,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 217–225.
[10] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in Proc. Int. Conf. Med. Image Comput.-Assist. Intervent. Cham, Switzerland: Springer, 2015, pp. 234–241.
[11] Y. Yu, Z. Gong, P. Zhong, and J. Shan, “Unsupervised representation learning with deep convolutional neural network for remote sensing image classification,” in Proc. Int. Conf. Image Graph. Cham, Switzerland: Springer, 2017, pp. 97–108.
[12] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” 2016, arXiv:1607.06450. [Online]. Available: https://arxiv.org/abs/1607.06450
[13] Y. Wu and K. He, “Group normalization,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Sep. 2018, pp. 3–19.
[14] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” 2016, arXiv:1607.08022. [Online]. Available: https://arxiv.org/abs/1607.08022
[15] V. Dumoulin, J. Shlens, and M. Kudlur, “A learned representation for artistic style,” 2016, arXiv:1610.07629. [Online]. Available: https://arxiv.org/abs/1610.07629
[16] X. Huang and S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 1501–1510.
[17] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” 2018, arXiv:1809.11096. [Online]. Available: https://arxiv.org/abs/1809.11096
[18] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4401–4410.
[19] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of styleGAN,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 8110–8119.
[20] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, “Modulating early visual processing by language,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 6594–6604.
[21] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, “Adaptive batch normalization for practical domain adaptation,” Pattern Recognit., vol. 80, pp. 109–117, Aug. 2018.
[22] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “FiLM: Visual reasoning with a general conditioning layer,” 2017, arXiv:1709.07871. [Online]. Available: https://arxiv.org/abs/1709.07871
[23] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. Vries, A. Courville, and Y. Bengio, “Feature-wise transformations,” Distill, vol. 3, no. 7, p. e11 Jul. 2018. [Online]. Available: https://distill.pub/2018/feature-wise-transformations
[24] Q. Yang, J. Huang, and W. Lin, “SwapText: Image based texts transfer in scenes,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 14700–14709.
[25] Y. Gong, L. Deng, Z. Ma, and M. Xie, “Generating text sequence images for recognition,” Neural Process. Lett., vol. 51, no. 2, pp. 1–12, 2020.
[26] F. Zhan, H. Zhu, and S. Lu, “Spatial fusion GAN for image synthesis,” in Proc. ICDAR2017 Competition on Multi-Lingual Scene Text Detection and Script Identification. Accessed: Apr. 30, 2020. [Online]. Available: https://tiecv.cvc.uab.es/?eh=8&om=interroduction
[27] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, “COCO-Text: Dataset and benchmark for text detection and recognition in natural images,” 2016, arXiv:1601.07140. [Online]. Available: https://arxiv.org/abs/1601.07140
[28] Y. Zhang, L. Gueguen, I. Zharkov, P. Zhang, K. Seifert, and B. Kadlec, “Uber-Text: A large-scale dataset for optical character recognition fromstreet-level imagery,” in Proc. Scene Understan, Workshop, 2017, pp. 1–2.
[29] A. Gupta, A. Vedaldi, and A. Zisserman, “Synthetic data for text localisation in natural images,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 2315–2324.
[30] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, “Generative visual manipulation on the natural image manifold,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 597–613.
[31] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, “Image inpainting for irregular holes using partial convolutions,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 2298–2304, Nov. 2017.
[32] C.-H. Lee, Z. Liu, L. Wu, and P. Luo, “MaskGAN: Towards diverse and interactive facial image manipulation,” in Proc. Adv. Neural Inf. Process. Syst., vol. 39, no. 11, 2020.
[33] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable network for image-based sequence recognition and its application to scene text recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 11, pp. 2298–2304, Nov. 2017.
[34] A. Hore and D. Zou, “Image quality metrics: PSNR vs. SSIM,” in Proc. 20th Int. Conf. Pattern Recognit., Aug. 2010, pp. 2366–2369.
ARSENY NERINOVSKY was born in 1996. He received the B.Sc. degree in information security and the M.Sc. degree in machine learning and data science from ITMO University, Saint Petersburg, Russia, in 2018 and 2020, respectively. He is currently a Deep Learning Researcher at the VK Saint Petersburg, working at applied deep learning technologies. His primary research interests include computer vision and AR technologies.

IGOR BUZHINSKY was born in 1992. He received the B.Sc. and M.Sc. degrees in applied mathematics and computer science from ITMO University, Saint Petersburg, Russia, in 2013 and 2015, respectively, the second M.Sc. degree in software engineering and service design from the University of Jyväskylä, Jyväskylä, Finland, in 2015, and the D.Sc. (Tech.) degree from Aalto University, Espoo, Finland, in 2019. He is a former Research Fellow with ITMO University and a former Postdoctoral Researcher with Aalto University. His research interests include formal verification, synthesis of finite-state models, and reliability of deep learning.

ANDREY FILCHENKOV (Member, IEEE) received the Specialist degree in computer science from St. Petersburg State University, in 2010, and the Ph.D. degree in computer science from Samara State Aerospace University, in 2013. From 2009 to 2014, he was a Researcher with the Saint Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences. From 2010 to 2014, he worked as a Teaching Assistant with Saint Petersburg State University. Since 2014, he has been heading the Machine Learning Laboratory, ITMO University, Saint Petersburg. He is currently an Associate Professor and is heading the Master Program “Machine Learning and Data Science”. He is also the author of more than 200 articles. His research interests include automated machine learning, image processing and generation, natural and programming language processing, and reinforcement learning application to various domains, including the routing problem.

Dr. Filchenkov is also a Laureate of the Ilya Segalovich Prize for the development of the scientific community and training of young scientists in 2019.