Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Магістерська робота

освітній ступінь – магістр

на тему: «**ГЕНЕРУВАННЯ ЗОБРАЖЕНЬ НОМЕРНИХ ЗНАКІВ МЕТОДАМИ ГЛИБИННОГО НАВЧАННЯ**»

Виконав: студент 2-го року навчання,

Освітньої програми «Прикладна математика», 113

Марчук Владислав Сергійович

Керівник Швай Н.О.,_________

кандидат фіз.-мат. наук, доцент

Рецензент ________________________
(прізвище та ініціали)

Магістерська робота захищена
з оцінкою ________________________

Секретар ЕК ______________________
«____» __________________ 20____ р.

Київ – 2023

CONTENTS

LIST OF ABBREVIATIONS

GAN	—	Generative Adversarial Network
CNN	—	Convolutional Neural Network
ALPR	—	Automatic License Plate Recognition
LP	—	License Plate
CycleGAN	—	Cycle-Consistent Adversarial Network
Pix2Pix	—	Image-to-Image Translation using Conditional Adversarial Networks
CUT	—	Contrastive Unpaired Translation
MLP	—	Multilayer Perceptron
PatchNCE	—	Patch-based Noise Contrastive Estimation
FastCUT	—	Fast Contrastive Unpaired Translation
GPU	—	Graphics Processing Unit
FID	—	Fréchet Inception Distance
CSV	—	Comma-Separated Values

INTRODUCTION

License plates play a crucial role in vehicle identification and tracking. They provide important information about the vehicle's registration and ownership. In recent years, the development of software products and technologies, such as deep learning, has revolutionized the field of computer vision and image processing. One promising application of deep learning is the generation of license plates using unpaired image-to-image translation techniques.

The traditional approach to license plate generation involves manually designing templates and labels, which can be time-consuming and resource-intensive. Additionally, gathering and labeling a large dataset of license plate images can be challenging and costly. Therefore, there is a need for an automated and efficient method to generate license plate images without relying on paired training data.

The main objective of this thesis is to develop a software application that utilizes the method of unpaired image-to-image translation for generating Ukrainian license plate images. To achieve this objective, the following research questions will be addressed:

- How can the method of unpaired image-to-image translation be adapted for generating Ukrainian license plates?
- What deep learning architecture and training procedure should be employed to achieve high-quality and realistic license plate generation?
- How does the generated output compare to real Ukrainian license plates in terms of visual quality and similarity?
- What are the limitations and potential challenges of the proposed approach?

The development of a software application that can automatically generate Ukrainian license plate images using unpaired image-to-image translation has several significant implications. Firstly, it eliminates the need for manual design and labeling of license plates, thereby saving valuable time and resources. Secondly, it opens up possibilities for various applications, including license plate synthesis for training machine learning models, data augmentation for computer vision tasks, and the creation of realistic virtual environments. Lastly, the use of unpaired image-to-image translation for license plate generation specifically for Ukrainian plates is a novel and unexplored area of research, providing an opportunity for pioneering contributions to the field.

By addressing these objectives and research questions, this thesis aims to contribute to the advancement of license plate generation techniques and demonstrate the potential of unpaired image-to-image translation in the context of Ukrainian license plates.

CHAPTER 1
LITERATURE REVIEW

1.1. Conditional generation

GAN-based methods have demonstrated remarkable results in generating high-fidelity images [1, 2]. However, the challenge of fully controlling the generated images through structured data is relatively under-explored. Historically, GANs have been primarily used in unsupervised settings, where the goal is to generate real and natural images. In its basic form, GANs consist of a generator that synthesizes images and a discriminator that classifies images as real or fake. Several approaches have enhanced GANs to incorporate supervised information.

One such approach is Conditional GAN (CGAN) [3], which extends traditional GANs to generate better images by including conditions on both the generator and the discriminator based on additional information, denoted as y. By conditioning on this additional information, CGAN can generate images that align with specific desired attributes or characteristics.

Another approach is AC-GAN [4], which expands the original goal of GANs to perform two tasks: classifying images as real or fake and predicting true attribute information represented by y. Unlike modeling the joint distribution as in CGAN, AC-GAN models the conditional distribution of the attribute y given the image x. This allows for more fine-grained control over the generated images based on specific attribute information.

Inspired by this strategy, conditional generation has been applied with various additional information types, including classes [5], text [6], bounding boxes [7], and pose estimations [8]. These additional inputs enable the generator to generate images that satisfy specific conditions or constraints.

One challenge encountered in conditional GANs is the topology mismatch between the latent space (input to the generator) and the output space (generated images). To address this issue, a proposed method [9] encourages a bi-Lipschitz mapping between the latent and output manifolds. This improvement enhances the quality of synthetic images in terms of image diversity and realism.

Numerous complex architectures have applied conditional generation and demonstrated good results in various domains, such as anime generation [10], brain metastases synthesis [11], 3D brain images at different stages of Alzheimer's disease [12], or gastritis image generation [13]. However, these previous works typically operate in a single-label setting, where the input consists of a label vector y and a random noise vector z. The discriminator produces two outputs: the probability of x being real or fake and the estimated conditional probability $p(x|y)$. While the use of the label vector y shows great potential for controlling the generated output, to the best of our knowledge, no existing work explores the possibility of gaining further control within a multi-label context.

1.2. License plate generation

In recent years, CNN-based Automatic License Plate Recognition (ALPR) methods [14] have achieved significant success. However, these methods often require a large amount of labeled data, which can be costly and time-consuming to collect. Additionally, existing ALPR systems may suffer from overfitting on over-represented characters or underperform on under-represented characters.

To address these limitations and improve ALPR systems, license plate (LP) image synthesis methods have been proposed to generate new examples of labeled training data [15]. These synthesized images can enhance the diversity and quantity of available data, leading to better performance.

One notable approach by Wang et al. [16] improves upon the CycleGAN-based model [17], which learns the mapping between synthetic (generated by a script) and real license plate images. This method leverages the power of generative adversarial networks to generate realistic LP images and enhance the training data for ALPR systems.

Similarly, Wu et al. [18] adopt a similar approach by improving the image-to-image based Pix2Pix generative architecture [19]. By training the model on pairs of synthetic and real LP images, they enhance the ability to generate realistic and diverse license plate images.

Although these methods have demonstrated good results, they have specific limitations. For example, the generated images may suffer from poor quality or lack variation, as they are often limited to single-image generation for a given license plate text input.

In contrast, Silvano et al. [20] propose a method to generate large volumes of accurate license plate images adhering to the Mercosur standard using a template-based synthesis approach. They incorporate data augmentation techniques that mimic real-life conditions such as artificial shading and sloping. This approach allows for the generation of license plate images with any desired text. However, this method heavily relies on a specific template and data augmentation, making it difficult to extend to other license plate standards or variations.

1.3. Contrastive Unpaired Translation

A significant advantage of Contrastive Unpaired Translation (CUT), compared to its competitors, lies in its ability to train and perform well even when both the input and output domains consist of only a single image [21]. This characteristic sets it apart from previous methods in the field. Furthermore, CUT introduces a novel approach by utilizing image patches instead of entire images during the translation process.

In CUT, the training procedure involves sampling negative patches, which are required to calculate the loss, from the same image rather than from other images in the dataset. Surprisingly, this sampling strategy yields superior results compared to scenarios where negatives are sampled from different images. This approach encourages the mapping of corresponding patches to the same location in relation to other patches (negatives), leading to improved output image quality. Additionally, this technique brings about notable advantages in terms of memory usage and the time required to train the model, resulting in more efficient training compared to the baseline method, which in this case is CycleGAN.

By focusing on image patches rather than entire images, CUT exploits the local characteristics and relationships within the input and output domains, enabling finer control over the translation process. This approach allows for more precise mapping of features and textures between the input and output images, leading to enhanced results.

The utilization of CUT in the context of Ukrainian license plate generation has several implications. It enables the model to learn and capture intricate details specific to license plates, such as characters, fonts, colors, and patterns, while maintaining the ability to generate high-quality and realistic images. Moreover, the use of image patches and the sampling strategy from the same image facilitate the preservation of spatial relationships and consistent mappings, resulting in visually coherent and accurate license plate images.

Overall, the employment of Contrastive Unpaired Translation for Ukrainian license plate generation brings notable advantages in terms of its ability to handle single-image domains, its focus on image patches for enhanced control, improved output image quality, reduced memory usage, and efficient training compared to the baseline method. These characteristics make CUT a promising approach for generating Ukrainian license plate images using unpaired image-to-image translation.

CHAPTER 2
METHODOLOGY

2.1. Explanation of the "Contrastive Unpaired Translation" method

Image-to-image translation aims to transform images from one domain to another while preserving the content of the original image. The goal is to change the style or specific characteristics of the images while maintaining the underlying information. This process can be seen as a disentanglement problem, where the content, which needs to be preserved, is separated from the appearance, which should be altered. Figure 2.1 provides an example of successful translation, showcasing a network trained on photographs of Parisian streets as the input domain and images depicting the canals of Burano as the output domain.

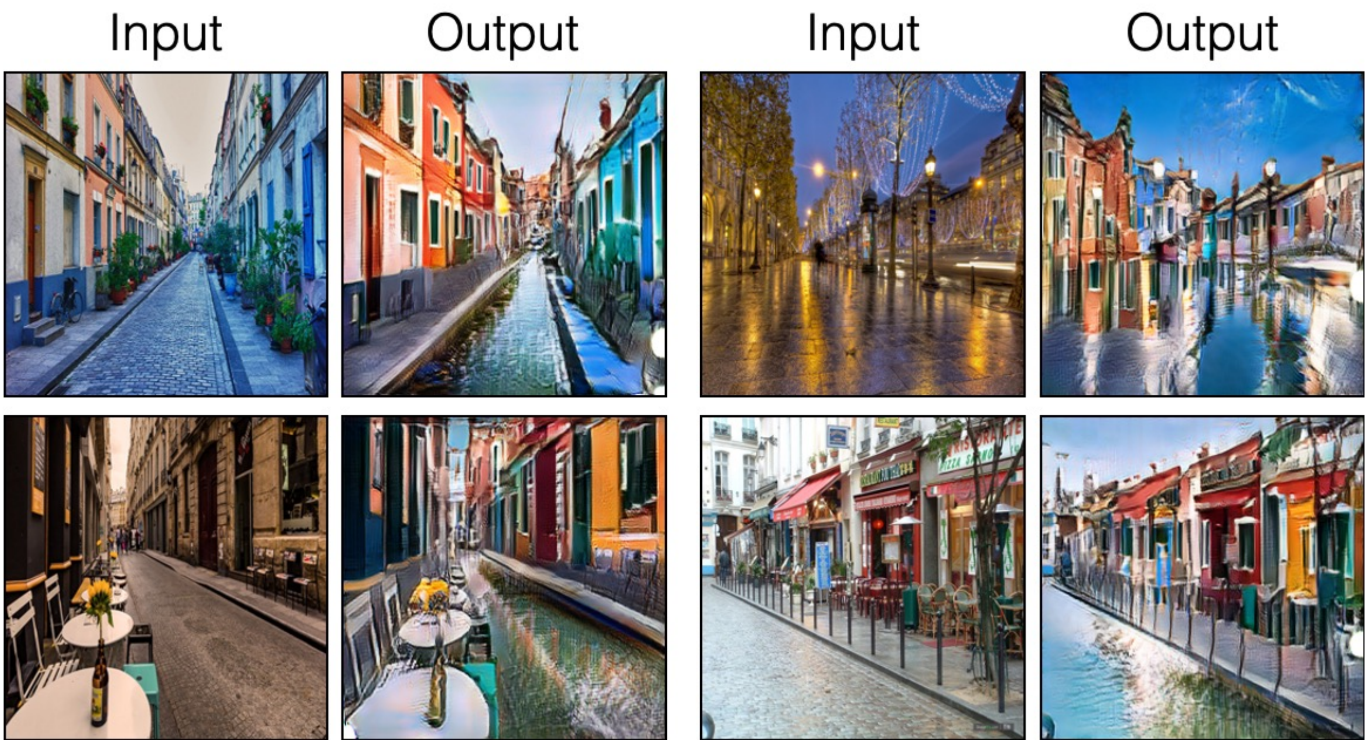


Fig. 2.1. Example of Image-to-Image Translation achieved by the CUT model - Parisian streets translated to depict the canals of Burano in Venice. Source: [21]

The model, known as "Contrastive Unpaired Translation" (CUT), is specifically designed to handle unpaired image translation tasks efficiently and effectively.

CUT introduces a novel methodology that leverages the power of deep learning to learn the mapping between two image domains without relying on paired training data. Instead, it takes advantage of unpaired datasets, where corresponding images between the input and output domains are not explicitly aligned. This flexibility allows for greater scalability and eliminates the need for labor-intensive manual annotation or data collection efforts.

The core principle behind CUT is to encourage the mapping of corresponding patches from the input and output domains to the same location in relation to other patches, while also optimizing the overall quality of the generated images. By focusing on image patches rather than entire images, CUT enables finer control over the translation process, preserving local characteristics and relationships between the domains.

Throughout the upcoming chapters, we will delve into the architectural details of CUT and the specific techniques it employs to achieve successful unpaired image translation. These discussions will encompass various components, such as the network architecture, loss functions, and training strategies employed by CUT to achieve high-quality and coherent translations between the Ukrainian license plate input domain and the synthesized output domain.

2.2. Architecture of the deep learning model used

The Contrastive Unpaired Translation model is employed to achieve the translation between the input domain X and the output domain Y, where X represents the set of unpaired input instances $\{x \in X\}$, and Y represents the set of unpaired output instances $\{y \in Y\}$. The objective is to transform images from the input domain X, which is a subset of $R^{H\times W\times C}$, to resemble images from the output domain Y, which is a subset of $R^{H\times W\times 3}$, while preserving the content of the input domain.

Unlike other methods that employ inverse auxiliary generators and discriminators, the CUT model simplifies the training process, resulting in reduced training times. The generator function G is composed of two components: an encoder G_{enc} followed by a decoder G_{dec}. Together, they produce the output image $\hat{y} = G(x) = G_{dec}(G_{enc}(x))$.

The adversarial loss [22] is utilized to encourage the output image to visually resemble the images from the target domain. The adversarial loss is defined as follows:

$$\mathcal{L}_{\mathrm{GAN}}(\mathrm{G},\mathrm{D},\mathrm{X},\mathrm{Y}) = \mathbb{E}_{\mathrm{y}\sim\mathrm{Y}}\log\left(\mathrm{D}(\mathbf{y})\right) + \mathbb{E}_{\mathrm{x}\sim\mathrm{X}}\log\left(1-\mathrm{D}(\mathrm{G}(\mathbf{x}))\right) \qquad (2.1)$$

To achieve mutual information maximization, noise contrastive estimation [23] is employed. An embedding is learned to associate a "query" with its corresponding "positive" while disassociating the "query" from other points in the dataset referred to as "negatives". The query, positive, and N negatives are mapped to K-dimensional vectors $\boldsymbol{v}, \boldsymbol{v}^{+} \in \mathbb{R}^{K}$, and $\boldsymbol{v}^{-} \in \mathbb{R}^{N\times K}$, respectively. These vectors are normalized onto a unit sphere to prevent the space from collapsing or expanding. An $(N+1)$-way classification problem is set up, where the distances between the query and other examples (positive and negatives) are scaled by a temperature $\tau = 0.07$ and passed as logits. The loss function is calculated using cross-entropy loss, as represented in the original CUT model [21]:

$$\ell(v, v^{+}, v^{-}) = -\log\left[\frac{\exp\left(v\cdot v^{+}/\tau\right)}{\exp\left(v\cdot v^{+}/\tau\right)+\sum_{\mathrm{n}=1}^{\mathrm{N}}\exp\left(v\cdot v_{n}^{-}/\tau\right)}\right] \qquad (2.2)$$

The ultimate goal is to establish associations between the input and output data. The "query" corresponds to the output, while the positives and negatives represent the corresponding and non-corresponding inputs, respectively.

Notably, the objective is to achieve shared content not only between the entire images but also between corresponding patches within the image. This multi-layer, patch-based objective is accomplished using the encoder G_{enc}. The feature stack generated by G_{enc} provides access to layers and spatial locations representing patches within the input image. A two-layer MLP (Multilayer Perceptron) network H_l is applied to the feature maps of selected layers, resulting in a stack of features $\{z_l\}_L = \{H_l(G_{enc}^{l}(x))\}_L$, where G_{enc}^{l} denotes the output of the l-th chosen layer. These layers are indexed by $l \in \{1,2,\ldots,L\}$, and a spatial location is denoted by $s \in \{1, \dots, S_l\}$, where s represents a specific spatial location and S_l represents the number of spatial locations in each layer. The corresponding feature is denoted as $z_l^s \in \mathbb{R}^{C_l}$, while $z_l^{S \setminus s} \in \mathbb{R}^{(S_l - 1) \times C_l}$ represents the remaining features, where C_l is the number of channels in each layer. A similar encoding process is applied to the output image ŷ, resulting in $\{\hat{z}_l\}_L = \left\{ H_l \left(G_{enc}^l \left(G_{dec} (G_{enc} (\boldsymbol{x})) \right) \right) \right\}_L$.

The objective is to match corresponding input-output patches at specific locations, utilizing other patches from the input image as negatives. PatchNCE [21] loss is employed to achieve this goal:

$$\mathcal{L}_{\mathrm{PatchNCE}}(\mathrm{G}, \mathrm{H}, \mathrm{X}) = \mathbb{E}_{\mathrm{x} \sim \mathrm{X}} \sum_{\mathrm{l}=1}^{\mathrm{L}} \sum_{\mathrm{s}=1}^{\mathrm{S_l}} \ell\left(\hat{\mathrm{z}}_{\mathrm{l}}^{\mathrm{s}}, \mathrm{z}_{\mathrm{l}}^{\mathrm{s}}, \mathrm{z}_{\mathrm{l}}^{\mathrm{S} \setminus \mathrm{s}} \right) \qquad (2.3)$$

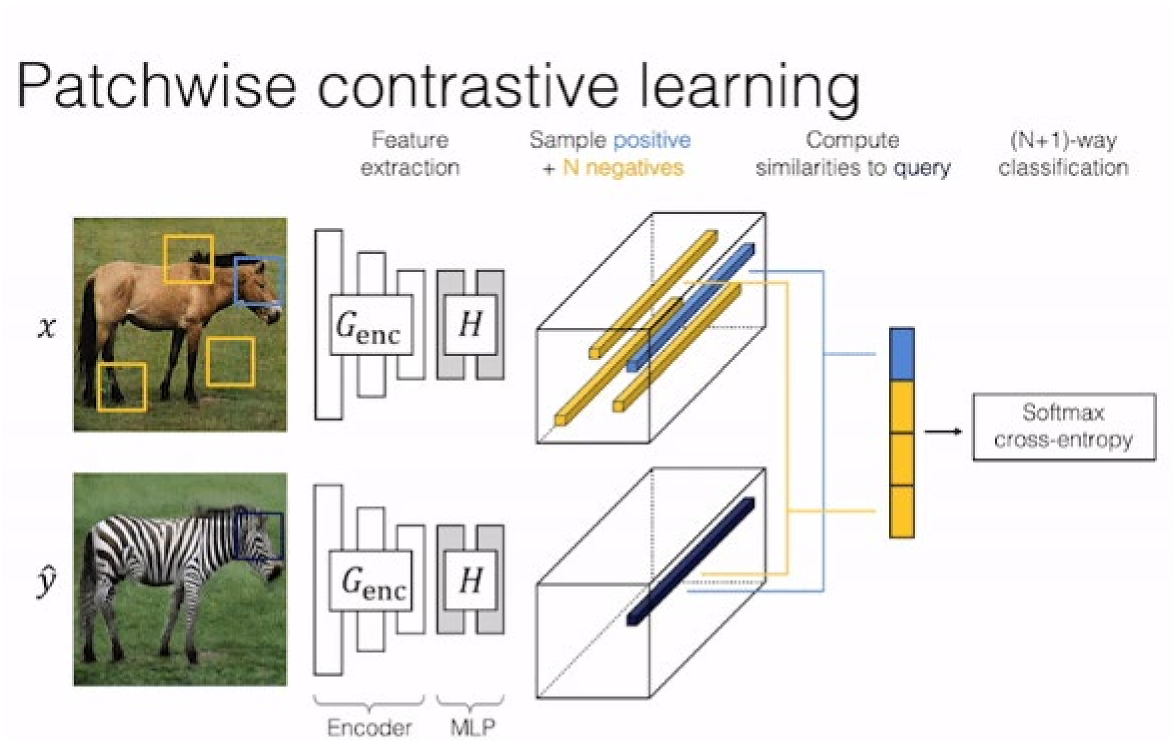

Patchwise contrastive learning
Feature extraction
Sample positive + N negatives
Compute similarities to query
(N+1)-way classification
x
ŷ
Genc
H
Softmax cross-entropy
Encoder
MLP

Fig. 2.2. Visual representation of Patchwise Contrastive Loss. Source: [21]

By combining the PatchNCE loss with the adversarial loss, the overall loss function is formulated as follows:

$$\mathcal{L}_{\mathrm{GAN}}(\mathrm{G}, \mathrm{D}, \mathrm{X}, \mathrm{Y}) + \lambda_{\mathrm{X}} \mathcal{L}_{\mathrm{PatchNCE}}(\mathrm{G}, \mathrm{H}, \mathrm{X}) + \lambda_{\mathrm{Y}} \mathcal{L}_{\mathrm{PatchNCE}}(\mathrm{G}, \mathrm{H}, \mathrm{Y}) \qquad (2.4)$$

where λ_X and λ_Y represent the weights assigned to the PatchNCE loss for domains X and Y, respectively.

In the case of the CUT model, $\lambda_X = \lambda_y = 1$. This identity loss is applied to prevent the generator from making unnecessary changes by calculating the PatchNCE loss between a translated image and a real image from the target domain dataset. Additionally, a lighter and faster variant called FastCUT is introduced, where $\lambda_X = 10$ to compensate for the absence of the regularizer (i.e., $\lambda_y = 0$). The FastCUT model is designed for scenarios with limited training time or GPU memory limitations, as it achieves satisfactory results similar to CycleGAN [24] while requiring significantly less memory and training time. However, it should be noted that the absence of the regularizer in FastCUT may lead to significant oscillations in the model's performance during training and potentially result in subpar final results.

2.3. Evaluation metric

To assess the quality of the generated images, the Frechet Inception Distance (FID) metric [25], [26] is employed. The FID metric quantifies the dissimilarity between the distribution of real images and generated images in a deep network space. The FID score is calculated using the following formula:

$$\mathrm{FID} = \|\mu_X - \mu_Y\|^2 + \mathrm{tr}\left(\Sigma_X + \Sigma_Y - 2\sqrt{(\Sigma_X\Sigma_Y)}\right), \qquad (2.5)$$

where μ_X and μ_Y represent the means of the real and generated image distributions, respectively, Σ_X and Σ_Y are their corresponding covariance matrices, tr denotes the trace operation, and $\sqrt{(\Sigma_X\Sigma_Y)}$ represents the matrix square root of the product of Σ_X and Σ_Y.

In essence, the FID metric estimates the distance between the distributions of real and generated images. A lower FID score indicates a higher level of similarity between the generated images and real images, signifying more convincing and realistic results.

By using the FID metric, the performance of the Contrastive Unpaired Translation (CUT) model can be quantitatively evaluated, providing an objective measure of the quality of the generated images. The goal is to achieve the lowest possible FID score, indicating that the generated images closely resemble the real images from the target domain.

2.4. Training details

This chapter outlines the training details of the CUT and FastCUT models, comparing them to the CycleGAN model. The initial CUT model configuration follows the settings presented in the Contrastive Learning for Unpaired Image-to-Image Translation paper [21]. The model includes a ResNet-based generator [27] with 9 residual blocks, a PatchGAN discriminator [28], Least SquareGAN loss [29], a batch size of 1, and an Adam optimizer [30] with a learning rate of 0.0002. These parameters are chosen to align with the original settings of CycleGAN [24]. The key difference is that the cycle-consistency loss in CycleGAN is replaced with the contrastive loss in CUT. For the CUT model, $\lambda_X = \lambda_y = 1$ is used in the loss function (equation 2.4), indicating equal weights for the PatchNCE loss on both domains. In the case of FastCUT, the loss function is modified to $\lambda_X = 10$ and $\lambda_y = 1$ to compensate for the absence of the regularizer.

The training process involves training each CUT experiment for 400 epochs. During the first 200 epochs, the learning rate remains constant, and then it gradually decays to 0 over the last 200 epochs. For the FastCUT model, training is conducted for 200 epochs, with the learning rate kept constant at 0.0002 for the first 150 epochs and then gradually decaying at a constant rate for the remaining 50 epochs. Additionally, the flip-equivariance augmentation, as described in the original paper [21], is applied during the training of the FastCUT model.

To calculate the PatchNCE loss, features are extracted from five different layers of the Genc. These layers correspond to the RGB pixels, the first and second downsampling convolutions, and the first and fifth residual blocks. The receptive field sizes for these layers are 1×1, 9×9, 15×15, 35×35, and 99×99, respectively. For each layer's features, a two-layer MLP is applied to 256 randomly sampled locations, resulting in 256-dimensional final features.

By training the models with these specific configurations and settings, the CUT and FastCUT models aim to achieve effective image translation performance while considering the limitations and computational efficiency compared to CycleGAN.

CHAPTER 3
SYSTEM DESIGN AND IMPLEMENTATION

3.1. Overview of the software application

The software application presented in this thesis is a PyTorch implementation designed for the generation of real and synthetic Ukraine license plate (LP) registration numbers. The application leverages a Generative Adversarial Network (GAN) model to generate realistic LP numbers with a certain level of distortions, mimicking the characteristics of real-life license plates.

The workflow of the application involves two main steps. First, synthetic Ukrainian LP numbers are generated using a predefined script. These synthetic LP numbers serve as input to the GAN model, which learns to generate realistic LP numbers that resemble the patterns and styles found on real license plates. The GAN model is trained using a combination of real and synthetic LP numbers, allowing it to learn the mapping between the synthetic and real images.

After the training phase, the GAN model is used for inference. Given a synthetic LP number as input, the model generates a corresponding real-life license plate number with a certain level of distortions. The distortions aim to simulate real-life conditions and variations observed on license plates, such as slight variations in font, spacing, and positioning.

To evaluate the performance of the generated license plates, an HTML file is generated as a result. This HTML file provides a visual comparison between the synthetic LP number, the generated real-life license plate, and an actual photograph of a real license plate. This comparison allows for a qualitative assessment of the realism and accuracy of the generated license plates.

3.2. System requirements and specifications

The software application has certain system requirements and specifications to ensure proper functionality. These requirements are outlined below:

- **prerequisites**: the application requires a Linux or macOS operating system. It is compatible with Python 3. Additionally, if GPU acceleration is desired, an NVIDIA GPU with CUDA CuDNN support is required;
- **framework and dependencies**: the application is built using PyTorch 1.1, a popular deep learning framework. Several dependencies are required for the application to run smoothly, including torchvision, visdom, dominate, and gputil. All the necessary dependencies are listed in the *requirements.txt* file, which can be used to install them;
- **hardware acceleration**: the software application has been tested and launched on Google Colab, utilizing hardware acceleration provided by an NVIDIA Tesla T4 GPU. This GPU acceleration significantly speeds up the computations, improving the overall performance of the application;
- **tensorflow compatibility**: although the software application primarily relies on PyTorch, it is compatible with Tensorflow as well. This compatibility allows users to take advantage of Tensorflow's GPU speedup over CPU, which can provide a speed boost of up to 35 times compared to CPU-only processing [31].

By meeting these system requirements and specifications, users can ensure a smooth and efficient operation of the software application. The compatibility with different operating systems, the utilization of GPU acceleration, and the inclusion of necessary dependencies contribute to the overall usability and performance of the application.

3.3. Description of the dataset used for training and evaluation

The software application utilizes a dataset consisting of both synthetic and real license plates for training and evaluation purposes. The dataset is described in detail below:

- **Synthetic license plates**: the synthetic license plates are generated using a Python script that reads registration numbers from a pre-defined CSV file. The generation process ensures that the synthetic license plates resemble real license plates, and it specifically uses the "Road UA" font [32] to achieve similarity with the fonts used on real license plates.
- **Real-life images**: the real license plate images are sourced from the online car market AUTO.RIA [33]. These images are downloaded and then processed by cropping them to focus on the license plate registration characters. Additionally, some transformations may be applied to the images, such as tilting, perspective warping, or darkening, to introduce variability and mimic real-world conditions.
- **License plate dimensions**: both the synthetic and real license plates in the dataset adhere to the dimensions of a regular single-line license plate [34], measuring 520 mm by 112 mm. This consistency ensures that the generated license plates match the physical dimensions of actual license plates.
- **Dataset split**: the dataset consists of a total of 100 images for training and 14 images for evaluation. The training images are divided into two folders: trainA and trainB. The trainA folder contains the synthetic license plates, while the trainB folder contains the corresponding real license plates. For evaluation purposes, the dataset includes two additional folders: testA and testB, which contain the synthetic and real license plates, respectively.

By combining synthetic and real-life license plates in the dataset, the software application can train and evaluate the GAN model effectively. The use of the Road UA font for synthetic plates, along with the inclusion of real-life license plate images, ensures that the generated license plates closely resemble real ones and exhibit the necessary variability for accurate evaluation and comparison.

3.4. Design principles and architecture

The software application follows a modular design with several scripts that serve specific functions. The design principles and architecture of the application are described as follows:

- **generate.py**: this script is responsible for generating synthetic license plates based on the information provided in a pre-defined CSV file;
- **make_dataset.py**: the purpose of this script is to organize the dataset for training. It copies the generated synthetic license plate images and the corresponding real license plate images into the training folders trainA and trainB, respectively. This organization enables the training process to access the necessary data for both synthetic and real license plates;
- **train.py**: this script is used to train the models, specifically CUT and FastCUT models. The training is performed based on the specified arguments, such as the *CUT_mode* argument, which determines the mode (CUT or FastCUT) for the training process. The script takes the training dataset from the specified *dataroot* (a root folder that contains trainA and trainB folders) and saves the outputs of the trained model under the specified *--name*;
- **test.py**: this script is used for conducting inference with the pretrained model. It allows the user to choose the pretrained model based on the *--name* argument and specify the mode (*--CUT_mode*) for the inference process (CUT or FastCUT). The script takes the test dataset from the specified *test_dataroot* (a root folder that contains testA and testB folders) and generates an HTML comparison file in the *result* folder. This HTML file provides a visual comparison between synthetic, generated, and real-life license plates.

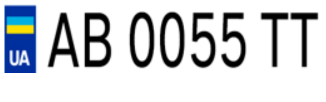
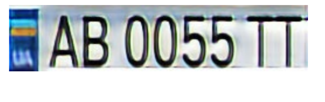
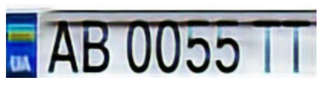
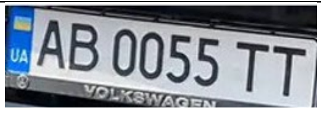
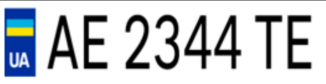
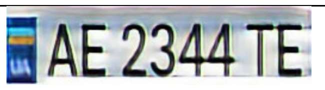
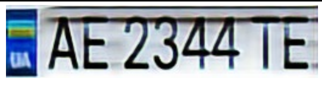
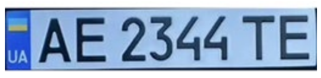
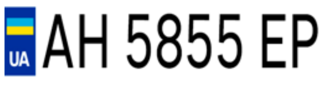
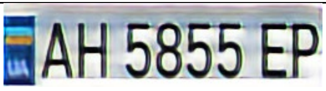
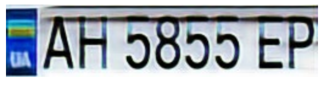
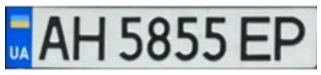
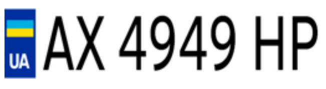
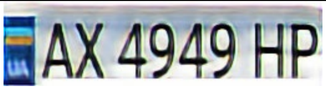
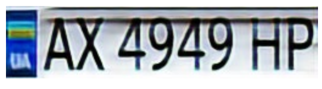
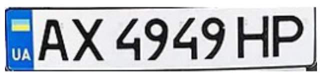
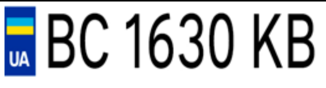
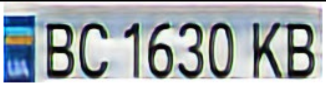
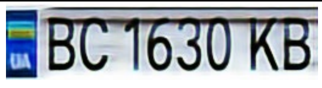
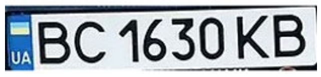
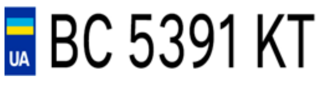
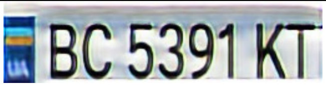
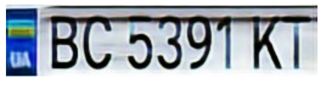
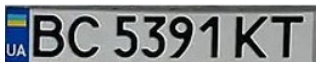
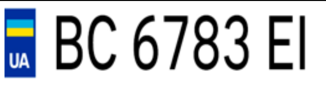
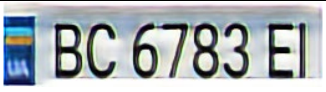
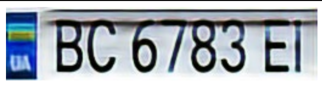
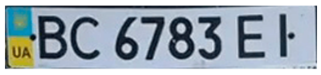
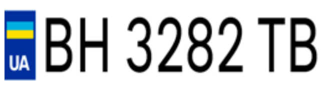
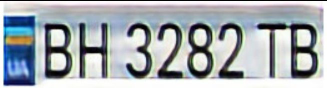
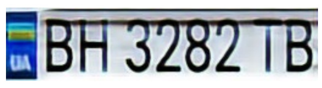
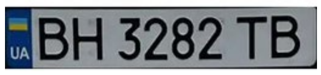
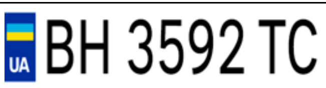
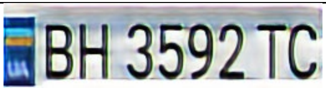
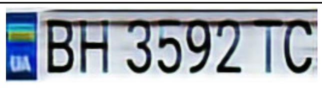
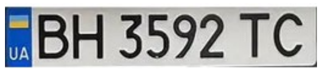
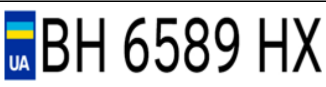
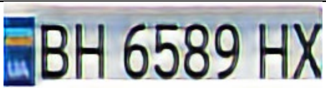
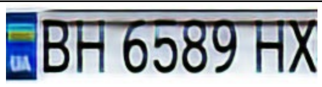
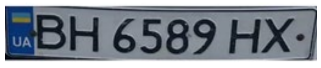
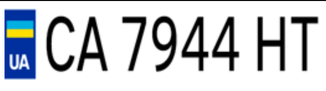
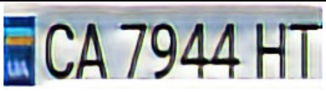
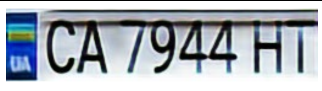
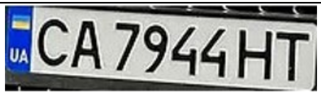
The architecture of the software application follows a pipeline-like approach. The generation of synthetic license plates is separated from the training and testing processes. The training phase utilizes the generated synthetic license plates and the corresponding real-life license plates to train the CUT and FastCUT models. The testing phase then employs the trained models to generate results and create a visual comparison in the form of an HTML file.

By following this design architecture, the software application enables the generation, training, and evaluation of synthetic and real-life license plates, allowing for the comparison of their visual quality and resemblance.

CHAPTER 4
RESULTS AND EVALUATION

4.1. Presentation of the generated license plates

The visual comparisons between synthetic, generated, and real license plates for both the CUT and FastCUT models are presented in the table below:

Synthetic	Generated FastCUT	Generated CUT	Real
			
			
			
			
			
			
			
			
			
			
			

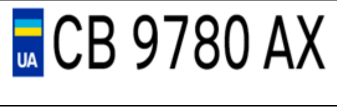
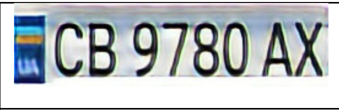
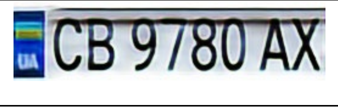
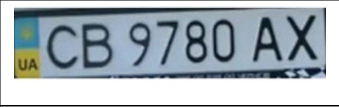
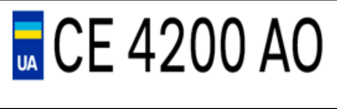
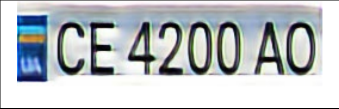
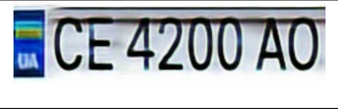
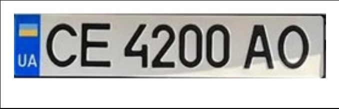
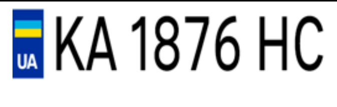
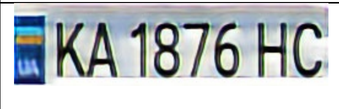
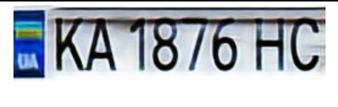
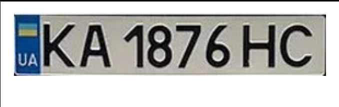
			
			
			

Table 4.1. Visual presentation of the generated license plates

Upon examining the generated license plates, it can be observed that there are differences between the results obtained from the CUT and FastCUT models. The generated license plates from the FastCUT model exhibit a less clear background and a more blurry font compared to those generated by the CUT model. This difference suggests that the FastCUT model may introduce more distortions in the generated license plates.

Furthermore, it is worth noting that the generated license plates from the CUT model exhibit an artifact in the form of a strip located above the characters. Unfortunately, no specific explanation is provided for this artifact, and further investigation is required to understand its origin and potential mitigation strategies.

The presented visual comparisons provide a comprehensive overview of the generated license plates, allowing for a visual assessment of their quality and resemblance to real license plates.

4.2. Qualitative evaluation of the generated plates

To evaluate the quality of the generated license plates, qualitative assessment was conducted.

The Frechet Inception Distance (FID) metric, as described in chapter 2.3, was used to evaluate the generated license plates. For the FastCUT model, the FID value obtained was 163.70, while for the CUT model, the FID value was 93.00. Because the FID scores can range from 0 to positive infinity, the license plates generated by the CUT model are 1.76 times more accurate in terms of resembling the real license plates compared to the FastCUT model.

The calculation speed of the FID metric was also measured for both the FastCUT and CUT models. The FastCUT model achieved a speed of 4.29 iterations per second, while the CUT model achieved a speed of 5.45 iterations per second. It should be noted that the calculation speed is dependent on the power of the GPU used for the evaluation.

Method	**FID**	**Sec/iter**
CUT	93.00	5.45
FastCUT	163.70	4.29

Table 4.2. Metric values for CUT and FastCUT models

Overall, the quantitative evaluation using FID scores indicates that the CUT model performs better in terms of generating license plates that closely resemble the real ones.

4.3. Discussion of limitations and challenges

Despite the promising results obtained in generating license plates using the CUT and FastCUT models, there are certain limitations and challenges that need to be addressed.

1. **Difficulty in capturing fine details**: one limitation is the difficulty in capturing fine details in the generated license plates. While the models are able to generate plates that bear a resemblance to real ones, they may struggle to reproduce intricate details accurately. Fine details such as small symbols, specific fonts, or subtle variations in color may not be captured with high fidelity. This limitation may affect the overall realism and authenticity of the generated license plates.
2. **Limited control over style transfer**: another challenge is the limited control over the style transfer process. The models aim to generate license plates that mimic the style of real plates while incorporating certain distortions. However, the level of control over the specific style elements and distortions may be limited. As a result, the generated plates may not precisely match the desired style or level of distortion. Improving the fine-grained control over the style transfer process could enhance the flexibility and customization options of the models.

3. **Dependency on training data quality and quantity**: the performance of the models is heavily dependent on the quality and quantity of the training data. The accuracy and diversity of the generated license plates are directly influenced by the characteristics of the training dataset. Insufficient or biased training data may lead to suboptimal performance and a limited range of generated plate variations. Additionally, the models may struggle to generalize well to unseen data if the training dataset does not adequately cover the full range of license plate styles and variations.

Addressing these limitations and challenges could involve exploring alternative model architectures, incorporating additional training techniques, or augmenting the training dataset with a wider variety of license plate styles and distortions. Furthermore, incorporating user feedback and iteratively refining the models based on real-world evaluations could help enhance their performance and address specific limitations.

CONCLUSION AND FUTURE WORK

In this thesis, we have explored the generation of license plates using the CUT and FastCUT models. The key findings and contributions of our research can be summarized as follows:

- We successfully developed a software application that generates synthetic license plate numbers and uses a Generative Adversarial Network model to create realistic license plates with distortions.
- The generated license plates were compared visually, and the results were presented in a table for both the CUT and FastCUT models. We observed that the FastCUT model produced license plates with a less clear background and more blurry font compared to the CUT model.
- The quality of the generated license plates was evaluated quantitatively using the Frechet Inception Distance metric. The FID scores for FastCUT and CUT were found to be 163.70 and 93.00, respectively. Based on these scores, the plates generated by CUT were found to be 1.76 times more accurate.

Considering the limitations and challenges discussed in chapter 4.3, there are several avenues for future research:

1. **Improving the capture of fine details**: further investigation and development of the models to enhance their ability to capture intricate details of license plates, such as small symbols and specific fonts, could lead to more realistic and accurate generated plates.
2. **Enhancing control over style transfer**: exploring techniques to provide users with more control over the style transfer process, allowing for finer adjustments to the desired style and level of distortion, would increase the flexibility and customization options of the models.
3. **Augmenting the training dataset**: expanding the training dataset to include a wider variety of license plate styles, distortions, and variations would help the models generalize better to unseen data and produce a more diverse range of generated plates.

The practical applications of the developed software application are numerous. It can be used for various purposes, such as generating synthetic license plates for testing and evaluation of Automatic License Plate Recognition systems, creating training data for computer vision models, or generating license plates for virtual environments in video games or simulations.

To further improve the system, potential enhancements include incorporating advanced image generation techniques, exploring alternative GAN architectures, and refining the training process by utilizing larger and more diverse datasets.

In conclusion, our research has demonstrated the feasibility of generating license plates using CUT and FastCUT models. Despite certain limitations, the models have shown promising results in generating realistic license plates. Future research and improvements in addressing the identified limitations will contribute to the advancement of license plate generation and its practical applications in various domains.

BIBLIOGRAPHY

1. Brock, A., Donahue, J., & Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In Int. Conference on Learning Representations (ICLR).

2. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2019). Analyzing and improving the image quality of StyleGAN. arXiv preprint arXiv:1912.04958.

3. Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

4. Odena, A., Olah, C., & Shlens, J. (2017). Conditional image synthesis with auxiliary classifier GANs. In Proc. of the 34th International Conference on Machine Learning-Volume 70 (pp. 2642–2651). JMLR.org.

5. Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., & Yosinski, J. (2017). Plug & play generative networks: Conditional iterative generation of images in latent space. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4467–4477).

6. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. N. (2017). StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In Proc. of the IEEE International Conference on Computer Vision (pp. 5907–5915).

7. Reed, S. E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., & Lee, H. (2016). Learning what and where to draw. In Advances in neural information processing systems (pp. 217–225).

8. Tang, W., Li, T., Nian, F., & Wang, M. (2018). MSCGAN: Multi-scale conditional generative adversarial networks for person image generation. arXiv preprint arXiv:1810.08534.

9. Ramasinghe, S., Farazi, M., Khan, S. H., Barnes, N., & Gould, S. (2021). Rethinking conditional GAN training: An approach using geometrically structured latent manifolds. Advances in Neural Information Processing Systems, 34.

10. Hamada, K., Tachibana, K., Li, T., Honda, H., & Uchida, Y. (2018). Full-body high-resolution anime generation with progressive structure-conditional generative adversarial networks. In Proc. of the European Conference on Computer Vision (ECCV) (pp. 0–0).

11. Han, C., Murao, K., Noguchi, T., Kawata, Y., Uchiyama, F., Rundo, L., Nakayama, H., & Satoh, S. (2019). Learning more with less: conditional pggan-based data augmentation for brain metastases detection using highly-rough annotation on MR images. arXiv preprint arXiv:1902.09856.

12. Jung, E., Luna, M., & Park, S. H. (2021). Conditional GAN with an attention-based generator and a 3D discriminator for 3D medical image generation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 318–328). Springer.

13. Togo, R., Ogawa, T., & Haseyama, M. (2019). Synthetic gastritis image generation via loss function-based conditional pggan. IEEE Access, 7, 87448–87457.

14. Španhel, J., Sochor, J., Juránek, R., Herout, A., Maršík, L., & Zemčík, P. (2017). Holistic recognition of low quality license plates by CNN using track annotated data. In IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) (pp. 1–6). IEEE.

15. Wu, C., Xu, S., Song, G., & Zhang, S. (2018). How many labeled license plates are needed? In Chinese Conference on Pattern Recognition and Computer Vision (PRCV) (pp. 334–346). Springer.

16. Wang, X., Man, Z., You, M., & Shen, C. (2017). Adversarial generation of training examples: applications to moving vehicle license plate recognition. arXiv preprint arXiv:1707.03124.

17. Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proc. of the IEEE international conference on computer vision (pp. 2223–2232).

18. Wu, S., Zhai, W., & Cao, Y. (2019). Pixtextgan: structure aware text image synthesis for license plate recognition. IET Image Processing, 13(14), 2744–2752.

19. Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proc. of the IEEE conference on computer vision and pattern recognition (pp. 1125–1134).

20. Silvano, G., Ribeiro, V., Greati, V., Bezerra, A., Silva, I., Endo, P. T., & Lynn, T. (2021). Synthetic image generation for training deep learning-based automated license plate recognition systems on the Brazilian Mercosur standard. Design Automation for Embedded Systems, 25(2), 113–133.

21. Park, T., Efros, A. A., Zhang, R., & Zhu, J. (2020). Contrastive learning for unpaired image-to-image translation. CoRR, abs/2007.15651. Retrieved from https://arxiv.org/abs/2007.15651

22. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems (Vol. 27). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

23. van den Oord, A., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. CoRR, abs/1807.03748. Retrieved from http://arxiv.org/abs/1807.03748

24. Zhu, J., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. CoRR, abs/1703.10593. Retrieved from http://arxiv.org/abs/1703.10593

25. Seitzer, M. (2020). pytorch-fid: FID Score for PyTorch. Retrieved from https://github.com/mseitzer/pytorch-fid

26. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), Advances in Neural Information Processing Systems (Vol. 30). Curran Associates, Inc. Retrieved from

https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf

27. Johnson, J., Alahi, A., & Li, F. (2016). Perceptual losses for real-time style transfer and super-resolution. CoRR, abs/1603.08155. Retrieved from http://arxiv.org/abs/1603.08155

28. Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. (2017). Image-to-image translation with conditional adversarial networks. Retrieved from https://arxiv.org/pdf/1611.07004.pdf

29. Mao, X., Li, Q., Xie, H., Lau, R. Y. K., & Wang, Z. (2016). Multi-class generative adversarial networks with the L2 loss function. CoRR, abs/1611.04076. Retrieved from http://arxiv.org/abs/1611.04076

30. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Retrieved from http://arxiv.org/abs/1412.6980

31. Google Colab. GPU Notebooks. Retrieved from https://colab.research.google.com/notebooks/gpu.ipynb

32. Road UA Font. Retrieved from https://agentyzmin.github.io/Road-UA-Font/

33. Auto.RIA.com. Retrieved from https://auto.ria.com

34. Vehicle registration plates of Ukraine. In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Vehicle_registration_plates_of_Ukraine