UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

BACHELOR THESIS No. 20

CONTRASTIVE LEARNING FOR IMAGE-TO-IMAGE TRANSLATION

Bernard Spiegl

Zagreb, June 2021

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

BACHELOR THESIS No. 20

CONTRASTIVE LEARNING FOR IMAGE-TO-IMAGE TRANSLATION

Bernard Spiegl

Zagreb, June 2021

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 12 March 2021

BACHELOR THESIS ASSIGNMENT No. 20

Student:	**Bernard Spiegl (0036514442)**
Study:	Electrical Engineering and Information Technology and Computing
Module:	Computing
Mentor:	prof. Siniša Šegvić

Title: **Contrastive learning for image-to-image translation**

Description:

Image-to-image translation is an important task of computer vision with many interesting applications. Recently, very interesting results have been achieved with convolutional models based on cyclic adversarial loss. A recent improvement of this approach learns from image crops and uses negative samples from the input image. In order to complete the thesis, the candidate has to choose a framework for automatic differentiation and get acquainted with libraries for handling matrices and images. Study and briefly describe existing approaches to image classification. Select a freely available dataset and partition it into train, validation and test subsets. Propose a suitable deep architecture for image-to-image translation with patchwise contrastive loss. Establish procedures for training models and validating hyperparameters. Evaluate the trained models and present the achieved accuracy. The thesis should be accompanied with the source code of the developed solution, complementary test sequences and results, as well as with necessary explanations and documentation. Please cite the used literature and document the received help.

Submission date: 11 June 2021

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 20

Pristupnik:	**Bernard Spiegl (0036514442)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Kontrastno učenje modela za prevođenje slika**

Opis zadatka:

Prevođenje slika važan je zadatak računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme vrlo zanimljive rezultate postižu konvolucijski modeli utemeljeni na kružnom suparničkom gubitku. Nedavno se pojavio postupak koji taj pristup poboljšava učenjem na isječcima te uzorkovanjem negativa iz ulazne slike. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće pristupe za klasifikaciju slike. Odabrati slobodno dostupni skup slika te oblikovati podskupove za učenje, validaciju i testiranje. Predložiti prikladnu arhitekturu dubokog modela za prevođenje slika. Uhodati postupke učenja modela i validiranja hiperparametara. Primijeniti naučene modele te prikazati i ocijeniti postignutu točnost. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 11. lipnja 2021.

I would like to thank Prof. Siniša Šegvić for provided insights and guidance during the making of this thesis.

I would also like to thank my sister Miheala for letting me use some of her art in the experimental phase. :-)

Finally, I would like to thank my family and friends for being supportive throughout my life.

CONTENTS

1. Introduction

In the recent years, deep convolutional networks and deep learning methods have become increasingly popular for tackling a wide variety of problems. As the demand for different types of data used for training deep convolutional models is continuously on the rise, significant efforts are being put into methods that would allow us to generate completely new, synthetic data from pre-existing data. One particularly interesting method is image-to-image translation which aims to take images from one domain and translate them so they have the characteristics of another domain whilst retaining the contents of the original images.

The thesis will cover *Contrastive Unpaired Image-to-Image Translation* [20], a successor to the *CycleGAN* [26] method from 2017. This new method harnesses the use of generative adversarial networks [5] while both improving on training speed and accuracy on certain problems in comparison to other baseline methods (e.g. *CycleGAN* [26], *MUNIT* [9], *DRIT* [16], *DistanceGAN* and *SelfDistance* [1], and *GCGAN* [4]). Generative adversarial networks [5] have proven to be especially useful in the domain of unsupervised learning i.e. when the data used to train the model is not labeled, which is the exact case in this kind of scenario.

The implementation details of this particular approach and what differentiates it from its predecessors will be discussed in more detail in the upcoming chapters. The basics of deep learning and neural networks are covered in chapter 3. In chapter 4 *Contrastive Unpaired Image-to-Image Translation* [20] is explained with the chapter 5 showing different experiments and results with proposed modifications to the existing models.

2. Related Work

Generative models. Ever increasing demand for data has lead to various methods for synthetic data generation. Most popular such models that have shown the best results are Generative Adversarial Networks (GANs) [5], Variational Autoencoders (VAEs) [14, 24] and, more recently, models based on normalizing flows [15, 23]. This thesis will be focusing on Generative Adversarial models with a specific approach to achieving image-to-image translation.

Image-To-Image Translation. A lot of recently developed models have focused on image-to-image translation, i.e. to learn the mapping between an input and an output image in a way that the content of the original input image is preserved while characteristics of output image are applied. This includes *CycleGAN* [26], *pix2pix* [10], *MUNIT* [9], *DRIT* [16], *DistanceGAN* and *SelfDistance* [1], *GCGAN* [4], etc. More detailed comparisons and explanations of different implementations available to date can be found in *Image-to-Image Translation: Methods and Applications* [19].

Contrastive Learning. For quite some time cross-entropy loss was the main loss function, especially in the supervised setting. However, recently a new type of loss known as *contrastive loss* has lead to state of the art performance both in supervised [12] and unsupervised [2] settings. This type of loss allows the model to learn an embedding where associated signals (in our case patches of an image) are brought together while signals that are not associated to each other are pushed further apart in the embedding space[1].

[1] a relatively low-dimensional space into which high-dimensional vectors are translated

3. Brief introduction to Deep Learning

3.1. Deep Learning

Deep learning is a branch of machine learning mainly focused on utilizing algorithms named artificial neural networks as they were inspired by the functioning of human brain. The term "deep" stems from the architectural nature of these networks as input is passed through multiple layers before reaching the output layer. Deep neural networks have applications in many fields such as natural language processing, speech recognition, computer vision, bioinformatics, etc. This thesis will be focusing on their application in computer vision.

3.2. Artificial Neural Network (ANN)

Artificial Neural Network (ANN) is an artificial intelligence algorithm which vaguely resembles the functioning of human brain and its neurons.

The neural network consists of an input and an output layer with an arbitrary number of interconnected hidden layers in-between. Each hidden layer contains neurons connected to the neurons of the previous and the following layer with each connection containing its own weight w_{ij}^{l} (Figure 3.1). Before the outputs of one layer are passed to another an activation function (e.g. Sigmoid, ReLU, tanh, etc.) is usually applied which introduces non-linearity to the outputs. This is done in order to enable the network to learn and perform more complex tasks.

During a forward pass, which results in a prediction, an input vector x is passed through the network and by performing series of linear and non-linear transformations an output is calculated (Equation 3.4).

$$x^{T} = [x_1, x_2, x_3, \dots x_n] \tag{3.1}$$

$$\mathrm{y}^0 = \mathrm{x} \tag{3.2}$$

$$\mathrm{W}^\mathrm{l} = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix} \tag{3.3}$$

$$\mathrm{y}^\mathrm{l} = \sigma(\mathrm{W}^\mathrm{l} \cdot \mathrm{x}^{\mathrm{l}-1}) \tag{3.4}$$

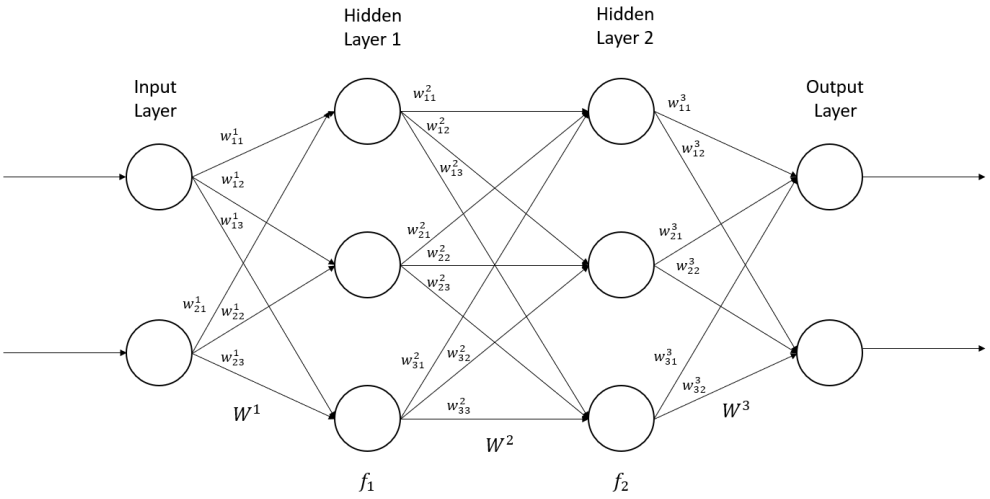


Figure 3.1: Simplified scheme of an **Artificial Neural Network (ANN)** containing 2 hidden, fully connected[1] layers, each comprised of 3 neurons.

3.2.1. Training process

In order to train the neural network we define a loss (cost) function $\mathcal{L}(y_{\text{true}}, y_{\text{pred}})$ which tells us how far the predicted (output) value is from the true value that the network is expected to output. During the training our goal is to minimize the loss function. This is done by adjusting the weights of each layer through a process called backpropagation.

Each forward pass computes an output used to calculate the loss. Having obtained the loss, gradient of the loss function (partial derivatives with respect to each of the

[1] a type of layer in which the neurons from each layer are connected to all the neurons in the following layer

weights) is calculated. This tells us how much each of the weights contributes to the total error. Hereafter, we perform backpropagation by moving backwards, layer by layer, through the network and adjusting each weight accordingly depending on the value of its partial derivative.

3.2.2. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of neural network that utilizes convolutional and pooling layers.

Convolutional layer is the key component of a CNN. They contain filters (kernels) with adjustable and learnable parameters. The filter convolves over the input producing a feature map as an output (Figure 3.3).

Pooling layer contains a window similar to a convolution filter that moves over the input and produces an output by applying a specified operation (e.g. max pooling, average, etc.) on sections of an input (Figure 3.2). The goal of such layers is to reduce the dimensionality of an input without losing too much information.

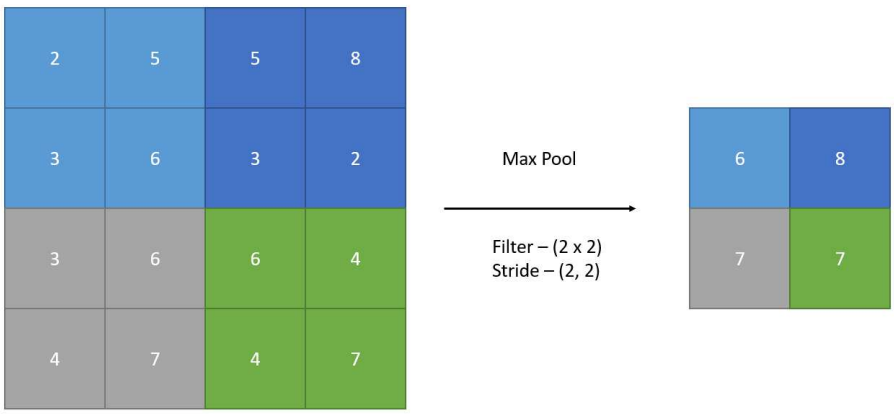
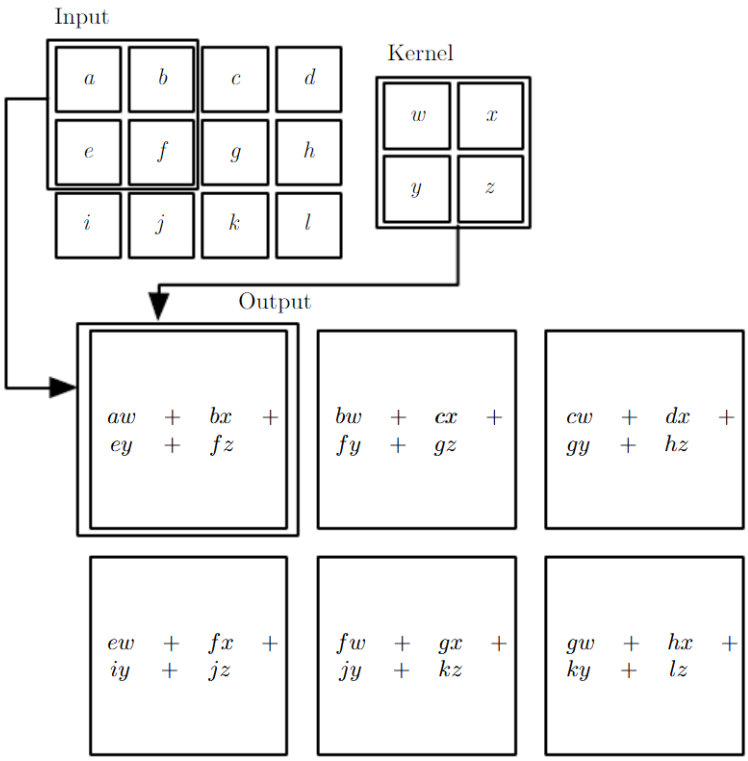
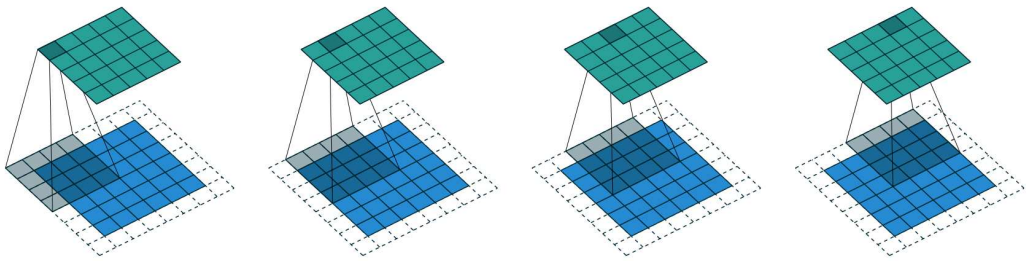


Figure 3.2: Example of a max pool operation on a 4×4 input with a filter size 2×2 and a stride $(2, 2)$. The filter moves over the input in steps of 2, both horizontally and vertically, and outputs the maximum of each section.



(a) Convolution output computation example. A set of dot products between each section and the kernel is calculated producing a feature map which is passed as an output. **Source:** [6]



(b) An example of a 4×4 kernel convolving over a 6×6 input padded with a 1×1 border. **Source:** [3]

Figure 3.3: Visual representation of a convolution.

3.2.3. Residual Neural Network (ResNet)

Residual Neural Network (ResNet) [7] is a neural network containing skip connections between nonadjacent hidden layers allowing us to train a deeper network. The skip connections (Figure 3.4) are added to mitigate the vanishing gradient problem[2] which often occurs in very deep models and can prevent the network from progressing further during its training.

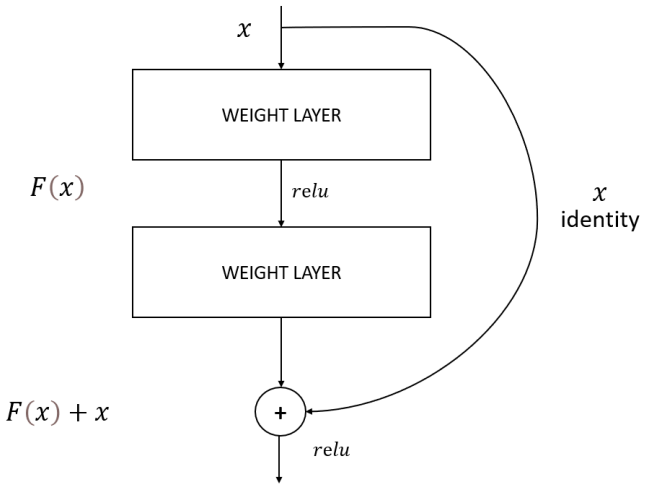


Figure 3.4: Skip connection in **Residual Neural Network (ResNet)**. The input x is passed through a set of transformations, i.e. weight layers, to produce the output. However, before applying the activation function and passing the output to the next layer, the previous input is added. The identity mapping ensures that the dimensions of the output and the previous input are matching.

3.2.4. Generative Adversarial Network (GAN)

Generative Adversarial Network (GAN) [5] is a neural network model based on adversarial process in which two networks are trained simultaneously.

The model consists of a generative network *G* and discriminative network *D*. During the training *G* is trying to maximize the probability of *D* making a mistake. More intuitively, these two networks play a zero-sum minmax two-player game where generator *G* is trying to "fool" the discriminator *D* while discriminator *D* has to tell whether the evaluated data sample is real or synthesized. By repeating this process both the generator *G* and the discriminator *D* get better, one at generating synthesized data and the other at telling which input data is real and which is fake, respectively. Ideally,

[2] a problem which occurs when the gradients of a loss function approach zero

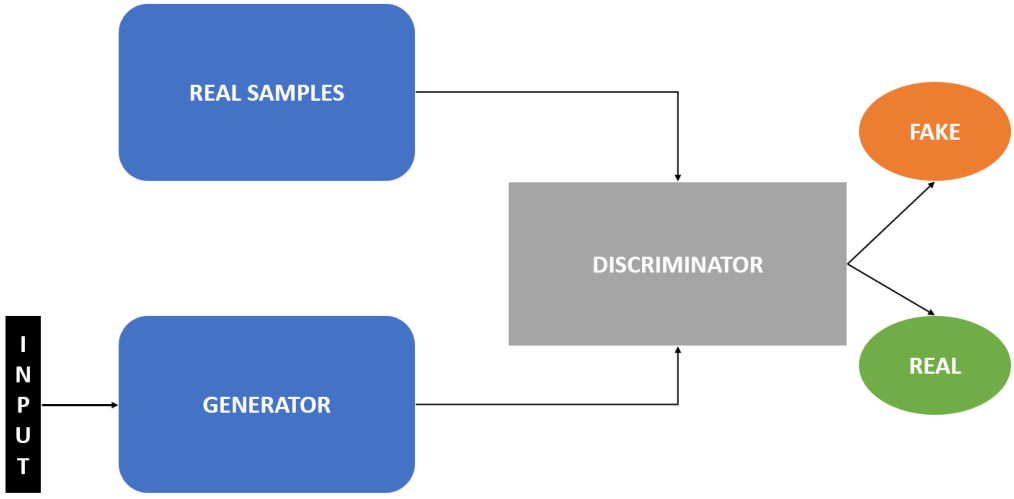


Figure 3.5: Structure of a **Generative Adversarial Network (GAN)**.

we want to reach a point where generated data, in our case images, looks deceptively convincing.

Unlike many other neural network models, where loss is very informative during the training and can reveal whether the training has converged or not, GANs require the use of alternative indicators such as IS (Inception Score) [21] and FID (Fréchet inception distance) [8] as the loss function tends to oscillate quite significantly during the training process (Figure 3.6).

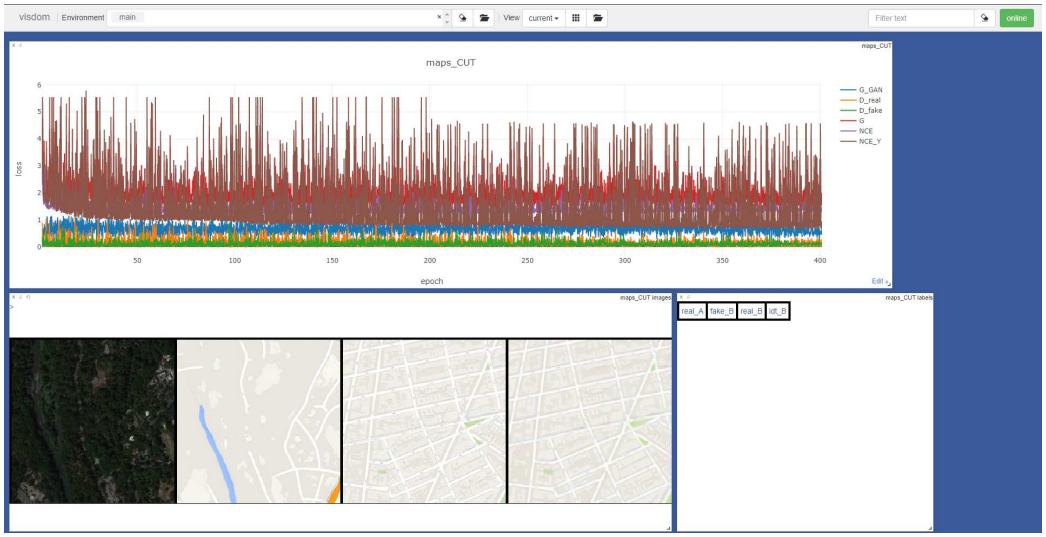


Figure 3.6: An example of oscillating loss function during the training of a GAN based **CUT model**.

3.3. PyTorch

PyTorch is an open source framework built for Python and primarily used for deep learning purposes. It enables straightforward creation of different deep neural model architectures.

A tensor is a fundamental unit of data used in deep learning. Therefore, PyTorch is optimized to leverage the capabilities of GPU in order to accelerate operations performed on tensors.

Another important feature of PyTorch is AutoGrad which gives it an ability to automatically compute a derivative of any expression. This significantly simplifies the process of backpropagation. By simply invoking *backward()* method, a backward pass is applied during which gradients are computed. The weights of the network are then adjusted by invoking the *step()* method.

The framework also has the ability to interoperate with NumPy which further expands its capabilities.

4. Contrastive Learning for Unpaired Image-to-Image Translation

Image-to-image translation aims to take images from one domain and translate them in such way that they adopt the style or certain characteristics of images from another domain while simultaneously preserving the contents of the original image. This can also be viewed as a disentanglement problem where the goal is to separate the content, which has to be preserved, from the appearance which is supposed to change.

An example of a successful translation can be seen in the Figure 4.1 where the network was trained using photographs of Parisian streets that represented the input domain and a set of images showing canals of Burano as the output domain.

The upcoming chapters *CUT* will review a model for performing image-to-image translation, including both its architectural details as well as the approach it takes to overcome the challenges of translation.

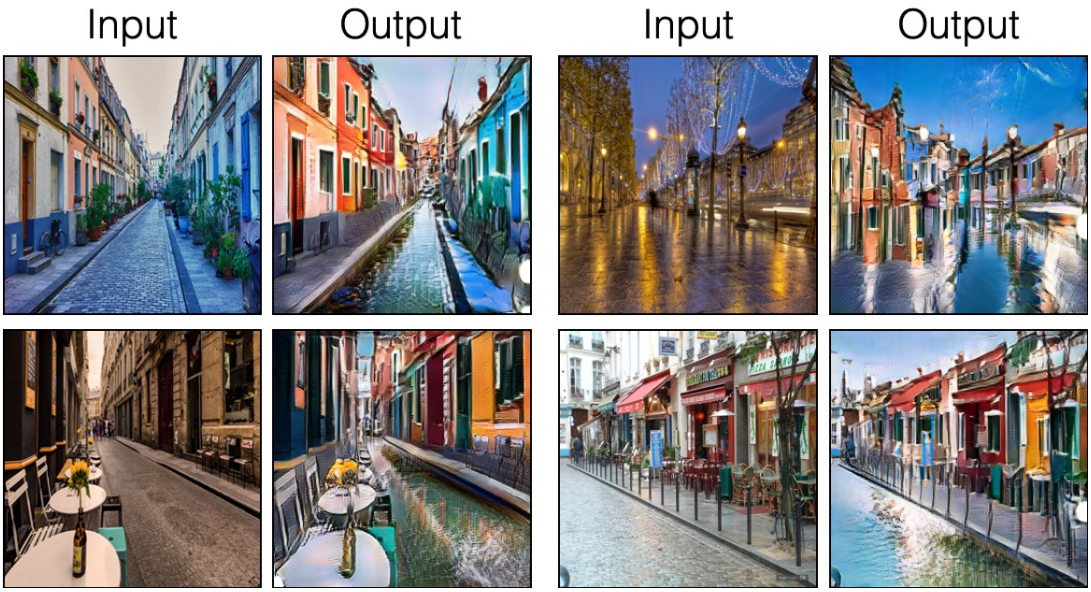


Figure 4.1: Example of **Image-to-Image Translation** achieved by the **CUT model** [20] - Parisian streets translated to depict the canals of Burano in Venice. **Source:** [20]

4.1. Contrastive Unpaired Translation

Contrastive Unpaired Translation (CUT) [20] is a newly proposed method for image-to-image translation and a direct successor to the well known *CycleGAN* [26]. However this method is a lighter and faster version meaning it requires less time to train while using less GPU memory in comparison to the *CycleGAN*.
However, instead of relying on cycle-consistency [26] to preserve the content, this method uses a new approach based on maximizing the mutual information between corresponding input and output patches. This approach both improves on the quality of the outputted image as well as the training times. The reason for this kind of approach is that the cycle-consistency [26] assumes the relationship between two domains is a bijection which often represents a problem as it is too restrictive. This is due to the fact that a perfect reconstruction, i.e. translation to the target domain and back to the original domain, is hard to achieve. Another significant benefit of this method is the ability to train and perform translations when both the source and output domains are only a single image. For example, given a realistic image as the input domain and a piece of art as the output domain, we are able to perform translation in such way that the realistic image preserves its original content while appearing as if it was created by the same artist that created a piece of art.

4.2. Methods

Given a dataset of unpaired instances $X = \{x \in \mathcal{X}\}$, $Y = \{y \in \mathcal{Y}\}$, our goal is to translate the images from the input domain $\mathcal{X} \subset \mathbb{R}^{H \times W \times C}$ to appear like images from the output domain $\mathcal{Y} \subset \mathbb{R}^{H \times W \times 3}$ while preserving content of the input domain.
Since this method avoids using inverse auxiliary generators and discriminators, the training procedure is significantly simplified, thus reducing the training times. The generator function *G* consists of two components, an encoder G_{enc} followed by a decoder G_{dec} which together produce the output image $\hat{\boldsymbol{y}} = G(\boldsymbol{x}) = G_{\text{dec}}(G_{\text{enc}}(\boldsymbol{x}))$.
In order to encourage the output to be visually as close as possible to the images contained in the target domain, **adversarial loss** [5] is used:

$$\mathcal{L}_{\text{GAN}}(G, D, X, Y) = \mathbb{E}_{\boldsymbol{y} \sim Y} \log(D(\boldsymbol{y})) + \mathbb{E}_{\boldsymbol{x} \sim X} \log(1 - D(G(\boldsymbol{x}))). \tag{4.1}$$

Mutual information maximization is achieved by using a noise contrastive estimation [25] and learning an embedding such that we closely associate a "query" and its "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 $\boldsymbol{v}^- \in \mathbb{R}^{N \times K}$ ($\boldsymbol{v}_n^- \in \mathbb{R}^K$ represents the n-th negative), respectively. Thereafter the vectors are normalized onto a unit sphere which prevents the space from collapsing or expanding. Afterwards an (N+1)-way classification problem is set up. The distances between the query and other examples (the positive and negatives) are scaled by a temperature $\tau = 0.07$ and passed as logits. The loss function is then calculated, which is represented as **cross-entropy loss** in the original *CUT* model [20].

$$\ell(\boldsymbol{v}, \boldsymbol{v}^+, \boldsymbol{v}^-) = -\log \left[\frac{\exp(\boldsymbol{v} \cdot \boldsymbol{v}^+ / \tau)}{\exp(\boldsymbol{v} \cdot \boldsymbol{v}^+ / \tau) + \sum_{n=1}^{N} \exp(\boldsymbol{v} \cdot \boldsymbol{v_n^-} / \tau)} \right]. \tag{4.2}$$

Later in chapter 5, the use of **focal loss** [17] is proposed as an improvement over the classic cross-entropy loss as it shows performance increase in the scenarios where the datasets are unbalanced.

The ultimate goal is to form association between the input and output data. The query refers to output whereas the positives and negatives are the corresponding and noncorresponding input.

It is important to notice that not only do we want whole images to share content but also the corresponding patches within the image. For instance, in the *Horse→Zebra*[1] example we should clearly be able to associate the specific body parts of a zebra to the specific body parts of an input horse more than to the other patches of the input image, e.g. a leg of a zebra should be more closely associated to the corresponding leg of a horse than to the other parts of its body. Moreover, colors of a zebra's body (black and white) are strongly associable to the color of a horse body rather than the background. Same analogy can be applied in the case of *Satellite→Maps* problem covered in chapter 5. Suppose we have a satellite image of an urban environment, after the translation is performed one should clearly be able to associate the streets seen in the *satellite* imagery with the streets visible in the *maps* representation, while disassociating them from other parts e.g. bodies of water, green surfaces, etc.

In order to enable achieving this type of correspondence a multi-layer, patch-based objective is set.

Role of the encoder G_{enc} is to perform the image translation meaning its feature stack

[1]The dataset consists of a test set containing 120 horse and 140 zebra images, and a train set containing 1067 horse images and 1334 zebra images, respectively. Both datasets are unpaired. Resolution of each image is 256×256.

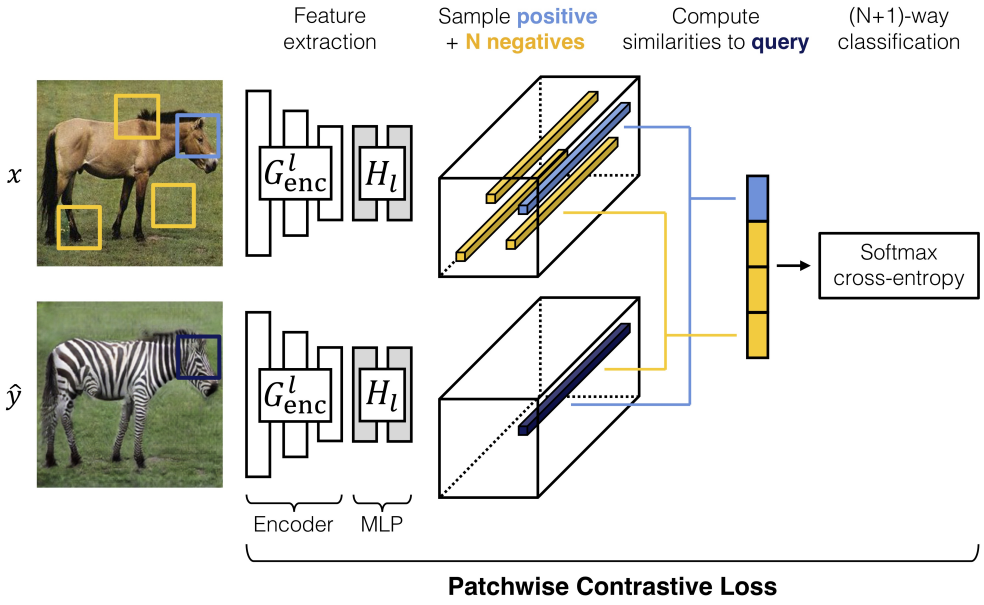


Figure 4.2: Visual representation of **Patchwise Contrastive Loss**. Images x and $\hat{y}$ are encoded into feature tensors. A query patch is sampled from the output $\hat{y}$ and compared to a patch (positive) from the input x at the corresponding location. By sampling additional N negatives at different locations in the input we form an (N+1)-way classification problem. The encoder part of the network G_{enc} is reused and a two-layer MLP network is added. We use the MLP to produce feature stacks by passing feature maps of layers through it. This procedure projects both the input and the output into a shared embedding space. **Source:** [20]

is at our disposal and we can make use of it. The feature stack contains layers and spatial locations representing patches of the input image, where deeper layers correspond to larger patches. L layers of interest are selected and their feature maps are passed through a two-layer *MLP* (*Multilayer Perceptron*) network H_l. This procedure produces a stack of features $\{z_l\}_L = \{H_l(G^l_{\text{enc}}(x))\}_L$, where G^l_{enc} represents the output of l-th chosen layer. Layers are then indexed into $l \in \{1, 2, ..., L\}$ and $s \in \{1, ..., S_l\}$ is denoted, where s represents a spatial location and S_l represents the number of spatial locations in each of the layers. Corresponding feature is referred to as $z^s_l \in \mathbb{R}^{C_l}$ while $z^{S\setminus s}_l \in \mathbb{R}^{(S_l-1)\times C_l}$ represents the rest of the features, where C_l is the number of channels at each layer. The output image $\hat{\boldsymbol{y}}$ is encoded into $\{\hat{z}_l\}_L = \{H_l(G^l_{\text{enc}}(G_{\text{dec}}(G_{\text{enc}}(\boldsymbol{x}))))\}_L$ in a similar manner.

Ultimately, our goal is to match corresponding input-output patches at a specific location. Other patches from within the input image are leveraged as negatives. As previously mentioned, we are aiming to achieve shared content not only between the images as a whole but also the corresponding patches within the image. To achieve this we use *PatchNCE* [20] loss:

$$\mathcal{L}_{\text{PatchNCE}}(G, H, X) = \mathbb{E}_{x \sim X} \sum_{l=1}^{L} \sum_{s=1}^{S_l} \ell(\hat{\boldsymbol{z}}_{\boldsymbol{l}}^{\boldsymbol{s}}, \boldsymbol{z}_{\boldsymbol{l}}^{\boldsymbol{s}}, \boldsymbol{z}_{\boldsymbol{l}}^{\boldsymbol{S} \setminus \boldsymbol{s}}). \tag{4.3}$$

By utilizing all the above mentioned, using Equation 4.1 and Equation 4.3, we form the following loss function:

$$\mathcal{L}(G, D, X, Y) + \lambda_X \mathcal{L}_{\text{PatchNCE}}(G, H, X) + \lambda_Y \mathcal{L}_{\text{PatchNCE}}(G, H, Y). \tag{4.4}$$

In addition to applying PatchNCE loss on images from domain $\mathcal{X}$ we also apply it on images from domain $\mathcal{Y}$ (referred to as *identity loss*) when using *Contrastive Unpaired Translation (CUT)* model where $\lambda_X = \lambda_Y = 1$. This is done in order to prevent the generator from making unnecessary changes - we calculate PatchNCE loss between a translated image and a real image taken from the target domain dataset. A lighter and faster method called *FastCUT* is also provided in which case $\lambda_X = 10$ in order to compensate for absence of the regularizer, i.e. for $\lambda_Y = 0$. In the experimental phase we show that although the model is faster and lighter to train, the absence of the regularizer can result in significant oscillations in model's performance during the training and potentially cause it to achieve subpar final results. *FastCUT* is designed for usage when time to train is restricted or GPU memory limitations are present as it can achieve satisfactory results similar to *CycleGAN* [26], while requiring significantly less memory and time to train.

5. Experiments & results

First section of this chapter covers the experiments and results done on *sat2map* [26] dataset and compares them to performance results of a pre-trained *CycleGAN* [26] model. Later on we move onto experiments done in a *single image translation* setting which are particularly interesting in situations when the available data is limited or, for instance, we want to apply a certain artistic style to a realistic image and vice versa.

5.1. CUT & FastCUT

5.1.1. Training details

This section covers the performance of *CUT* and *FastCUT* models with different losses against *CycleGAN* on *sat2map* dataset.

Sat2map dataset contains test, train and validation sets each containing 1098, 1096, and 1098 images, respectively. The dimensions of images are 600×600 with each image containing its representation in both domains, i.e. in input domain $\mathcal{A}$ and output domain $\mathcal{B}$. Although the images in the dataset are inherently paired, we preserve the nature of the unpaired approach as the images are not taken in corresponding pairs, i.e. the source and target datasets are unaligned, during the training process. Domain $\mathcal{A}$ consists of various satellite images, predominantly those containing dense urban areas, while domain $\mathcal{B}$ contains corresponding representations in the style of street maps.

In order to evaluate the quality of generated images we use **Fréchet Inception Distance (FID)** [22, 8] metric:

$$\text{FID} = \|\mu_X - \mu_Y\|^2 + \text{tr}\Big(\Sigma_X + \Sigma_Y - 2\sqrt{(\Sigma_X \Sigma_Y)}\Big). \tag{5.1}$$

In short, what this method does is empirically estimates the distribution of real and generated images in a deep network space and computes the divergence between them. Ideally, we want to have the lowest possible FID score, which indicates that the generated images are convincing.

The initial *CUT* model includes ResNet-based generator [11] with 9 residual blocks, PatchGAN discriminator [10], Least SquareGAN loss [18], batch size of 1 and Adam optimizer [13] with learning rate of 0.0002 - this is in-line with the original settings provided in *Contrastive Learning for Unpaired Image-to-Image Translation* [20] paper. The reason we choose such parameters is to be as close as possible to *CycleGAN* [26]. The only difference is is that the ℓ_1 cycle-consistency loss[1] ([26]) is replaced with contrastive loss [20].

As previously mentioned, in section 4.2 for *CUT* model we use $\lambda_X = \lambda_Y = 1$ whereas for *FastCUT* $\lambda_X = 10$ and $\lambda_Y = 0$ is used in the loss function (Equation 4.4).

Each *CUT* experiment is trained up to 400 epochs, where during first 200 epochs the learning rate is kept constant and throughout the last 200 epochs it gradually decays to 0. Moreover, *FastCUT* model is trained up to 200 epochs with first 150 epochs keeping a constant learning rate of 0.0002 and last 50 decaying it at a constant rate. Additionally, flip-equivariance augmentation is applied when training *FastCUT* as described in the original paper [20]. For calculating PatchNCE loss we extract features from 5 different layers of the G_{enc}. Namely, RGB pixels, the first and second downsampling convolution as well as the first and the fifth residual block. These layers correspond to receptive fields of sizes 1×1, 9×9, 15×15, 35×35 and 99×99. For each layer's features, a 2-layer MLP is applied onto 256 randomly sampled locations in order to acquire 256-dim final features.

[1] forward cycle-consistency loss: $x \to G(x) \to F(G(x)) \approx x$
backward cycle-consistency loss: $y \to F(y) \to G(F(y)) \approx y$

5.1.2. Experiments

For the first experiment we train the *CUT* model without any modifications using methodology explained above, we will refer to it as *CUT - CE*. While this model already surpasses the performance of *CycleGAN* [26] by ∼5% in terms of FID score (Table 5.1) and is quite good at translating satellite imagery containing exclusively urban environments (Figure 5.3), severe artifacts and completely incorrect translations can be seen in some edge cases, e.g. in images that contain bodies of water or green surfaces (Figure 5.4).

This is most likely caused by the imbalance of the dataset as it contains a plethora of images containing urban environments while it lacks satellite imagery of green surfaces, bodies of water and other similar edge cases that cause problems in translation.

Method	**FID** ↓	**sec/iter** ↓	**Mem(GiB)** ↓
CycleGAN	99.28	0.33	9.48
CUT - CE	94.48	0.3	3.39
CUT - Focal ($\gamma = 2, \alpha = 0.25$)	**91.44**	0.3	3.39
CUT - Focal ($\gamma = 3, \alpha = 0.5$)	93.24	0.3	3.39
FastCUT - CE	136.98	0.19	2.34
FastCUT - Focal ($\gamma = 2, \alpha = 0.25$)	**102.41**	0.19	2.34

Table 5.1: Comparison of different methods used on sat2map dataset. FID is calculated using fid-pytorch library [22]. It is calculated between domain $\mathcal{B}$ test dataset in *sat2map* and 500 generated images using a trained model. Testing was done using an NVIDIA Tesla P100 using Google Colab platform.

In order to alleviate this issue we introduce **focal loss** [17] (Figure 5.1):

$$\mathrm{FL}(p_t) = -\alpha(1-p_t)^{\gamma}\log(p_t) \tag{5.2}$$

This is done by replacing previously used cross-entropy loss within the *PatchNCE* loss while keeping other specified training parameters.

Focal loss allows us to penalize difficult, misclassified examples while significantly reducing the loss for well classified examples. Intuitively, this will allow our model to focus on edge case scenarios and adjust accordingly to increase its overall performance while making minimal changes in the scenarios where its performance is already good.

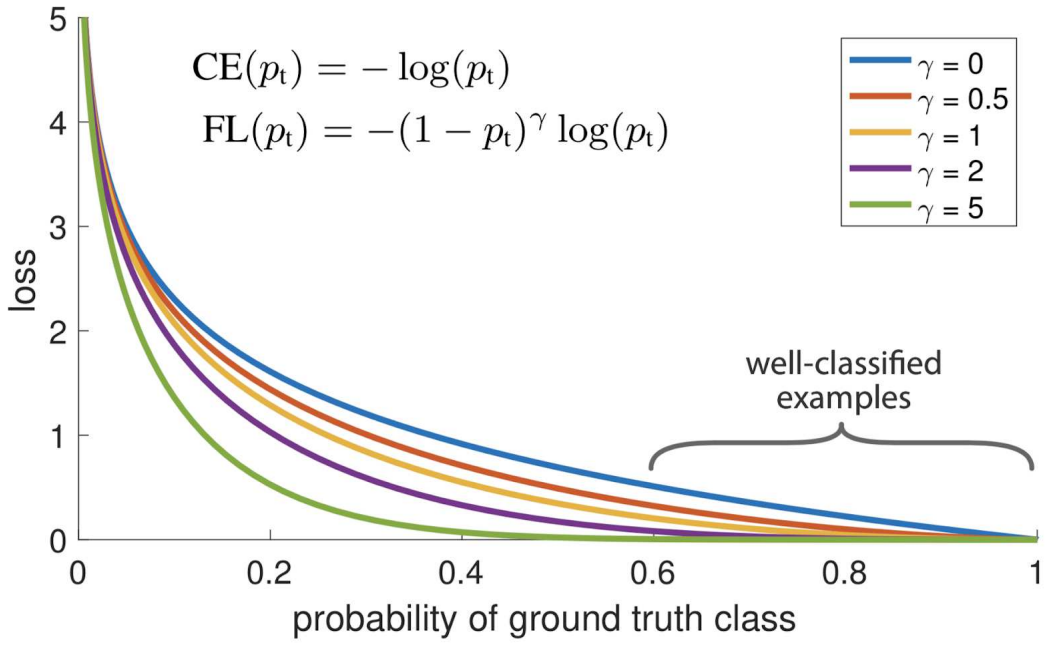


Figure 5.1: Visual representation of focal loss [17] depending on different values of γ. Focal loss with $\gamma = 0$ is a cross-entropy loss. **Source:** [17]

We proceed to train the model using focal loss with $\gamma = 2$ and $\alpha = 0.25$ and achieve additional $\sim$3% boost in FID score compared to regular cross-entropy model, bringing the total performance increase compared to *CycleGAN* up to $\sim$8%. An additional model is trained with $\gamma = 3$ and $\alpha = 0.5$, and, while it still outperforms *CycleGAN* and *CUT - CE*, it fails to outperform the initial configuration using $\gamma = 2$ and $\alpha = 0.25$.

It is clearly visible in Table 5.1 that all the tested *CUT* models use significantly less memory (almost $3\times$ less) than *CycleGAN*, train faster and simultaneously manage to outperform *CycleGAN's* FID scores.

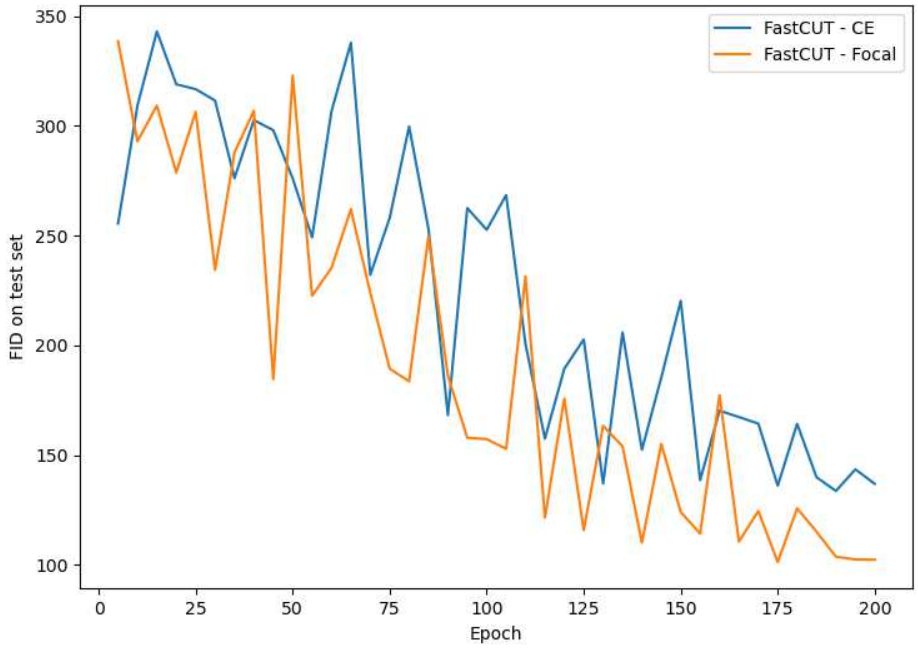


Figure 5.2: FID scores of *FastCUT* variants on test set each 5 epochs throughout 200 epochs. While the performance of both models oscillates due to the missing identity loss, version using focal loss converges faster reaching a lower final FID score.

Furthermore, same experiment is done on the *FastCUT* model. First its trained using Cross-Entropy loss (denoted as *FastCUT - CE*), and afterwards using focal loss. As shown in Table 5.1 focal loss version significantly outperforms cross-entropy version and even comes close to *CycleGAN* in terms of FID, all while consuming over $4\times$ less memory and taking $\sim$40% less time per iteration which brings down the total training time from $\sim$37 hours to just $\sim$11 hours.

Figure 5.3: Example of a successful *Satellite* $\rightarrow$ *Map* translation achieved by *CUT - CE* model (Target domain (domain $\mathcal{B}$) is represented by the image labeled *real_B* while the image labeled *real_A* represents the source domain (domain $\mathcal{A}$). The result of translation is shown in the middle image labeled *fake_B*)

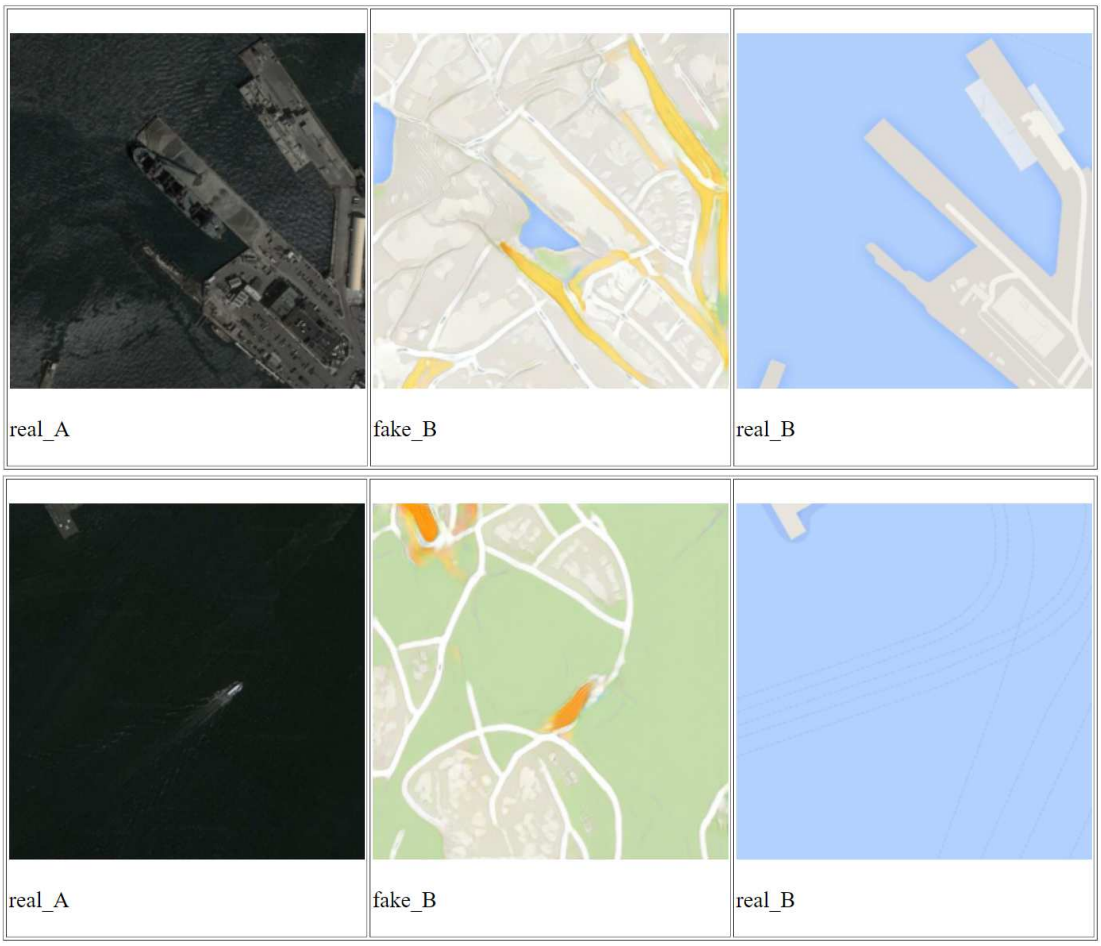


Figure 5.4: Example of *satellite* → *map* edge case translation achieved by *CUT - CE* model where the output is severely artifacted. (Target domain (domain $\mathcal{B}$) is represented by the image labeled *real_B* while the image labeled *real_A* represents the source domain (domain $\mathcal{A}$). The result of translation is shown in the middle image labeled *fake_B*)

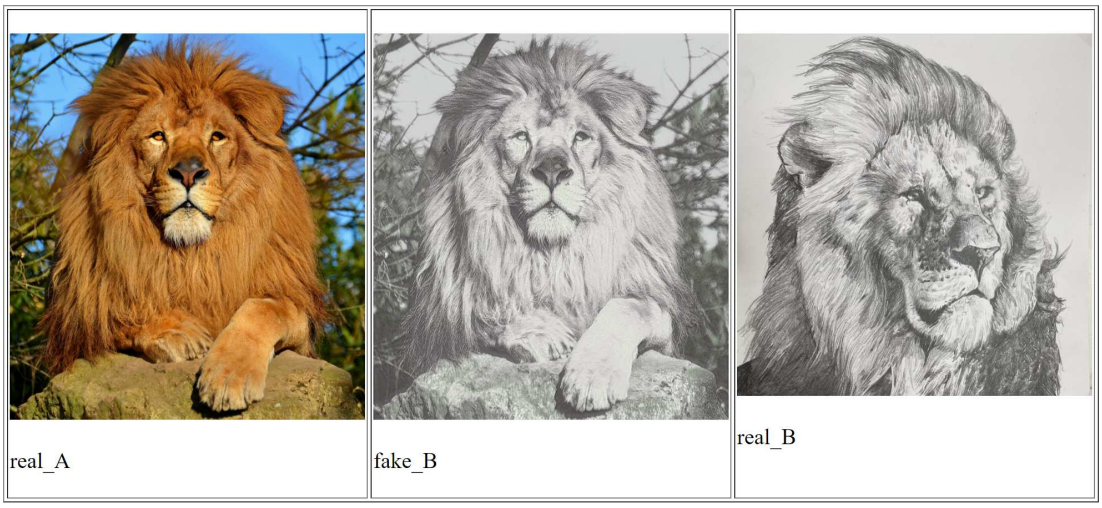
5.2. SinCUT

In this section we will cover conducted experiments in the scenario when both $\mathcal{A}$ and $\mathcal{B}$ domain are a single, high-res image using *SinCUT* model. The experiments are conducted using two completely different styles of photos as a target domain. Namely, one drawing is a photorealistic sketch while the other is a more complex and abstract piece of art. We pick the latter in order to see how well the model is able to capture specific features and characteristics of artist's style.

Each image is around the size of 1000×1000 and the model is trained on 16 random crops sized 128×128 in each iteration. This has to be done due to the fact that the whole image of that resolution cannot fit on a commercial GPU. Additionally, to avoid overfitting, the crops are divided into 64×64 tiles before being passed to the discriminator.

For the first example shown in Figure 5.5 the model was trained up to 100 000 iterations for each translation direction. We can see that the model performs much better when converting a real life photo to sketch (Figure 5.5a) than the other way around (Figure 5.5b).

In the second example the model is initially trained up to 1 epoch (100 000 iterations), same as in the first example, but then let to train for 16 epochs (1 600 000 iterations) with first 8 at a constant learning rate and last 8 decaying. While the model manages to preserve the content and adopts the style to some extent after 1 epoch (Figure 5.6a) it comes nowhere close to target domain in terms of level of detail and overall quality. In an attempt to circumvent this issue we train the model for full 16 epochs. In Figure 5.6b we clearly see that the model was able to pick up specific features of artist's style much better than in the first case. However, it only retains the outlines of the original image and almost entirely fails to preserve the original content.

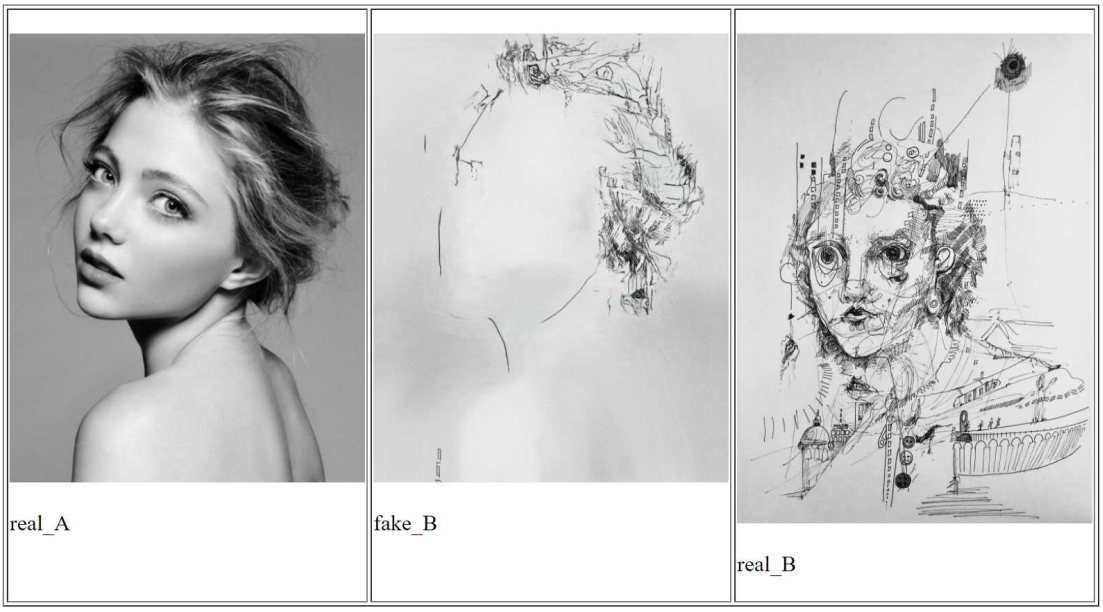


(a) Translation from real life photograph to sketch

(b) Translation from sketch to real life photograph

Figure 5.5: Example of photorealistic sketch $\leftrightarrow$ photograph translation (Target domain (domain $\mathcal{B}$) is represented by the image labeled *real_B* while the image labeled *real_A* represents the source domain (domain $\mathcal{A}$). The result of translation is shown in the middle image labeled *fake_B*)

(a) Result after 100 000 iterations (1 epoch)



(b) Result after 1 600 000 iterations (16 epochs)

Figure 5.6: Example of a portrait → abstract painting translation (Target domain (domain $\mathcal{B}$) is represented by the image labeled *real_B* while the image labeled *real_A* represents the source domain (domain $\mathcal{A}$). The result of translation is shown in the middle image labeled *fake_B*)

6. Conclusion

The main focus of this thesis was image-to-image translation by applying contrastive learning method. This approach focuses on maximizing mutual information between corresponding input and output patches in order to form the final output image which contains the content of input domain and the characteristics of output domain.

By conducting a series of experiments on *sat2map* dataset using different variations of the *CUT* model we compare their performances based on FID, training speeds and memory usage. Based on the results, an upgraded version of *CUT* model which uses focal loss instead of cross-entropy loss within *PatchNCE* loss is proposed as it alleviates the imbalance issue of *sat2map* dataset.

We also show the ability of this model to perform translations when both input and output domains are a single images. Through several experiments its performance and ability to capture specific features is tested in cases when the output domain is a photorealistic sketch and a more abstract piece of art, respectively.

Further improvements could be done in the single image translation domain in order to make the model perform better when the target domain is a more abstract piece of art.

BIBLIOGRAPHY

[1] Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/59b90e1005a220e2ebc542eb9d950b1e-Paper.pdf.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. URL https://arxiv.org/pdf/2002.05709.pdf.

[3] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018. URL https://arxiv.org/pdf/1603.07285.pdf.

[4] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, Kun Zhang, and Dacheng Tao. Geometry-consistent generative adversarial networks for one-sided unsupervised domain mapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/papers/Fu_Geometry-Consistent_Generative_Adversarial_Networks_for_One-Sided_Unsupervised_Domain_Mapping_CVPR_2019_paper.pdf.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.,

2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 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, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

[9] Xun Huang, Ming-Yu Liu, Serge J. Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 179–196, 2018. URL https://arxiv.org/pdf/1804.04732.pdf.

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei Efros. Image-to-image translation with conditional adversarial networks. pages 5967–5976, 07 2017. doi: 10.1109/CVPR.2017.632. URL https://arxiv.org/pdf/1611.07004.pdf.

[11] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016. URL http://arxiv.org/abs/1603.08155.

[12] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *CoRR*, abs/2004.11362, 2020. URL https://arxiv.org/abs/2004.11362.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-*

9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

[14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014. URL https://arxiv.org/pdf/1312.6114.pdf.

[15] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2020. ISSN 1939-3539. doi: 10.1109/tpami.2020.2992934. URL http://dx.doi.org/10.1109/TPAMI.2020.2992934.

[16] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. *CoRR*, abs/1808.00948, 2018. URL http://arxiv.org/abs/1808.00948.

[17] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. URL http://arxiv.org/abs/1708.02002.

[18] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016. URL http://arxiv.org/abs/1611.04076.

[19] Yingxue Pang, Jianxin Lin, Tao Qin, and Zhibo Chen. Image-to-image translation: Methods and applications. *CoRR*, abs/2101.08629, 2021. URL https://arxiv.org/abs/2101.08629.

[20] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. *CoRR*, abs/2007.15651, 2020. URL https://arxiv.org/abs/2007.15651.

[21] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. URL http://arxiv.org/abs/1606.03498.

[22] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.1.1.

[23] Moein Sorkhei, Gustav Eje Henter, and Hedvig Kjellström. Full-glow: Fully conditional glow for more realistic image generation. *CoRR*, abs/2012.05846, 2020. URL https://arxiv.org/abs/2012.05846.

[24] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder, 2021. URL https://arxiv.org/pdf/2007.03898.pdf.

[25] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

[26] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. URL http://arxiv.org/abs/1703.10593.

Contrastive Learning for Image-to-Image Translation

Abstract

Image-to-image translation models transfer images from input domain to output domain in an endeavor to retain the content of the image. Contrastive Unpaired Translation is one of the existing methods for solving such problems. Significant advantage of this method, compared to competitors, is the ability to train and perform well in cases where both input and output domains are only a single image. Another key thing that differentiates this method from its predecessors is the usage of image patches rather than whole images. It also turns out that sampling negatives (patches required to calculate loss) from the same image achieves better results than a scenario where the negatives are sampled from other images in the dataset. This type of approach encourages mapping of corresponding patches to the same location in relation to other patches (negatives) while at the same time improves the output image quality and significantly decreases memory usage as well as the time required to train the model compared to CycleGAN method used as a baseline. Through a series of experiments we show that using focal loss in place of cross-entropy loss within the PatchNCE loss can improve on the model's performance.

Keywords: deep learning, image-to-image translation, contrastive learning, convolutional neural networks, generative adversarial networks, image generation

Kontrastno učenje modela za prevođenje slika

Sažetak

Modeli za prevođenje slika pretvaraju sliku iz izvorne domene u odredišnu domenu i pri tome nastoje sačuvati sadržaj slike. Metoda s kontrastnim učenjem jedan je od pristupa za rješavanje ovakvog tipa problema. Prednost ove metode je što omogućuje treniranje u slučaju kada je domena samo jedna slika. Ono što ju razlikuje od prethodnika je činjenica da koristi isječke slika prilikom treniranja umjesto cijelih slika. Nadalje, pokazano je da uzimanje negativa (isječci slike potrebni za izračunavanje gubitka) iz iste slike postiže bolje rezultate nego kad se negativi uzimaju iz drugih slika u datasetu. Ovakav pristup potiče mapiranje odgovarajućih isječaka na približno isto mjesto u odnosu na druge isječke (negative) dok istovremeno poboljšava kvalitetu rezultirajuće slike i smanjuje količinu memorije kao i vrijeme treniranja u odnosu na baseline CycleGAN metodu. Kroz eksperimente također pokazujemo da korištenje žarišnog gubitka umjesto unakrsne entropije unutar PatchNCE gubitka dovodi do poboljšanja performansi modela.

Ključne riječi: duboko učenje, prevođenje slika, kontrastno učenje, konvolucijske neuronske mreže, generativne suparničke mreže, generiranje slika