Unsupervised Generation of Free-Form and Parameterized Avatars

Adam Polyak, Yaniv Taigman, and Lior Wolf, *Member, IEEE*

Abstract—We study two problems involving the task of mapping images between different domains. The first problem, transfers an image in one domain to an analog image in another domain. The second problem, extends the previous one by mapping an input image to a tied pair, consisting of a vector of parameters and an image that is created using a graphical engine from this vector of parameters. Similar to the first problem, the mapping's objective is to have the output image as similar as possible to the input image. In both cases, no supervision is given during training in the form of matching inputs and outputs.
We compare the two unsupervised learning problems to the problem of unsupervised domain adaptation, define generalization bounds that are based on discrepancy, and employ a GAN to implement network solutions that correspond to these bounds. Experimentally, our methods are shown to solve the problem of automatically creating avatars.

Index Terms—Deep Learning, Domain Adaptation, Neural Network, Cross-Domain Transfer, Analysis by Synthesis, Domain Transfer Network, Tied Output Synthesis.

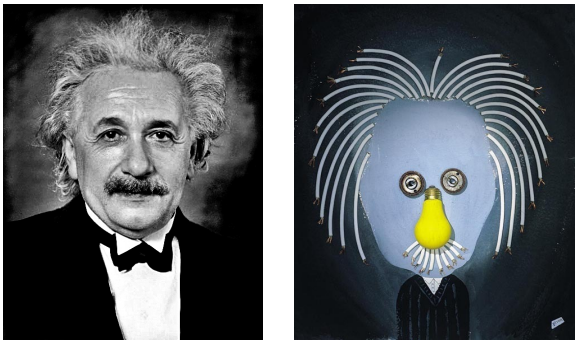
1 Introduction[1]

The artist Hanoch Piven creates caricatures by arranging household items and scrap material in a frame and photographing the result, see Fig. 1. In this work, we ask "how can a computer create such images?"

Given a training set consisting of Piven's images, Generative Adversarial Networks (GANs) can be used to create images that are visually similar to the training set. However, this is not enough. The generated image needs to preserve the identity of the input image. The mapper can, in principle, permute the identities and return an output image of Bob given an input image of Alice's face, with no effect on the GAN loss.

The Domain Transfer Network we present solves this, by utilizing a perceptual function f to learn a mapping $G : \mathcal{X} \rightarrow \mathcal{Y}$ such that $f(x) \sim f(G(x))$, where G is the GAN generative function, $\mathcal{X}$ is the source domain and $\mathcal{Y}$ is the target domain.

The architecture of the Domain Transfer Network includes the function G, which is a composition of the input function f and a learned function g. A compound loss that integrates multiple terms is used. One term is a GAN term that encourages the creation of samples that are indistinguishable from the training samples of the target domain. The second loss term enforces the perceptual similarity for every x in the source domain training set. The third loss term is a regularizer that encourages G to be the identity mapping for all $x \in \mathcal{Y}$.



(a) (b)

Fig. 1. (a) Formal portrait of Albert Einstein. (b) A caricature by Hanoch Piven.

Fig. 2. From the image on the top left, our method computes the parameters of the face caricature below it, which can be rendered at multiple views and with varying expressions by the computer graphics engine.

As we show, when applied to the problem of avatar creation, e.g., using a computer graphics emoji-generating API, this free-form solution produces highly identifiable and appealing images. However, these images are easily distinguishable (by humans) from avatars created by the computer graphics engine. Indeed,

- *A. Polyak is with the School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel and Facebook AI Research. This work was carried out in partial fulfillment of the requirements for his Ph.D. degree. E-mail: adampolyak@fb.com*
- *L. Wolf is with the School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel and Facebook AI Research. E-mail: wolf@fb.com*
- *Y. Taigman is with Facebook AI Research. E-mail: taigman@fb.com*

Manuscript received January 15, 2018; revised May 22, 2018.

1. This manuscript is based on previously published contributions [32], [37].

common sense tells us that for any reasonably sized training set, the generated images would be easily recognized by humans as being synthetic. We, therefore, ask a second question: "How do you incorporate this knowledge into the generation process?".

Assume that the virtual or the physical world in which the training images of the target domain were generated, is captured by an available engine e, which, given a vector of parameters u produces an image $e(u)$. In the case of the physical world, where Piven operates, e provides us with the ability to generate an image from the domain of Pivenesque images. Piven, of course, does not create random face images. He creates caricatures of famous people, preserving the identity information in the original image.

In the case of the virtual world, this process is analogous to generating computer avatars, based on the user's appearance, using a graphical engine. In order to allow the avatars to be easily manipulated, each avatar is represented by a set of "switches" (parameters) that select, for example, the shape of the nose, the color of the eyes and the style of hair, all from a predefined set of options created by artists. The visual appearance of the avatar adheres to a set of constraints, which are governed by a computer graphics engine that renders an image based on the set of parameters. Moreover, once this set is determined, the avatar can be rendered in many variations (Fig. 2).

Therefore, we present the problem of Tied Output Synthesis (TOS), which is learning to map an input image to two tied outputs: a vector in some parameter space and the image generated by this vector. While it is sufficient to recover just the vector of parameters and then generate the image, a non-intuitive result of our work is that it is preferable to recover the analog image first. In any case, the mapping between the input image and either of the outputs should be learned in an unsupervised way, due to the difficulty of obtaining supervised samples that map input images to parameterized representations. In avatar creation, it is time consuming for humans to select the parameters that represent a user, even after considerable training. The selected parameters are also not guaranteed to be the optimal depiction of that user. Therefore, using unsupervised methods is both more practical and holds the potential to lead to more accurate results.

The presented solution, Tied Output Synthesis Network, solves the problem presented above by augmenting the Domain Transfer Network with additional constraints. Given a domain, $\mathcal{X}$, an engine e and a function f, we would like to learn a generative function G, such that f is invariant under G and that for all samples $x \in \mathcal{X}$, there exists a configuration u such that $G(x) = e(u)$. Other than the functions f and e, the training data is unsupervised and consists of a set of samples from the source domain $\mathcal{X}$ and a second set from the target domain of e.

In addition to the practical motivation, humans can learn to create parameterized analogies without using matching samples. Understanding possible computational processes is, therefore, a novel AI objective. In addition to posing new computer vision and AI problems, we also place the TOS problem in the mathematical context of other domain shift problems. By doing so, we derive an appropriate generalization bound and develop an algorithm that matches the terms of the generalization bound.

2 Background

Generative Adversarial Networks GAN [10] methods train a generator network G that synthesizes samples from a target distribution, given noise vectors, by jointly training a second network d. G is trained jointly with a discriminator network d, which distinguishes between samples generated by G and a training set from the target distribution. The goal of G is to create samples that are classified by d as real samples. The specific generative architecture we employ is based on the architecture of [29]. Since the image we create is based on an input and not on random noise, our method is related to Conditional GANs, which employ GANs in order to generate samples from a specific class [25] and the work of [30] which generates images based on a text.

The recent work by [4], has shown promising results for learning to map embeddings to their pre-images, given input-target pairs. They also employ a GAN, as well as additional losses in the feature- and the pixel-space. Their method is able to invert the mid-level activations of AlexNet and reconstruct the input image. By contrast, in our Domain Transfer Network, we solve the problem of unsupervised domain transfer and apply the loss terms in different domains: pixel loss in the target domain, and feature loss in the source domain. Compared to our second method, which solves the Tied Output Synthesis problem, their method does not generate configurations.

The CoGAN method [20], like our Tied Output Synthesis method, generates a pair of tied outputs. However, this method generates the two outputs based on a random vector and not on an input image. More importantly, the two outputs are assumed to be similar and their generators (and GAN discriminators) share many of the layers. In our case, the two outputs are related in a different way: a vector of parameters and the resulting image. The solutions are also vastly different. CoGAN can solve the problem of free-form domain transfer. However, in [41], CoGAN was shown to fail in the task of cross domain image transfer.

A recent work, which studied the learning of 3D structure from images in an unsupervised manner, shares some of the computational characteristics with our problem [13]. The most similar application to our parameterized problem, involves a parametrization of a 3D computer graphics object with 162 vertices, each moving along a line, a black-box camera projecting from 3D to 2D and a set of 2D images without the corresponding 3D configuration. The system then learns to map 2D images to the set of vertices. This setting shares with us the existence of a fixed mapping from the vector of parameters to the image. In our case, this mapping is given as a neural network that will be termed e, in their case, it is given as a black box, which, as discussed in Sec. 7, is a solvable challenge. A more significant difference is that in their case, the images generated by the fixed mapping are in the same domain as the input, while in our case it is from a different domain. The method employed in [13] completely differs from ours and is based on sequential generative models [11].

Image synthesis with CNNs The supervised network of [5] receives as input a one-hot encoding of the desired model, as well as view parameters and a 3D transformation and generates the desired view of a 3D object as both an RGB image and a foreground/background map. DC-IGN [16] performs a similar task with less direct supervision. The proposed framework learns an encoder-decoder network, in which the representation at the middle bottleneck is disentangled by controlling the training of individual neurons and structuring the minibatches to explore different aspects of the view and illumination. The training set of this method is stratified but not necessarily fully labeled and is used to disentangle the image representation in an encoder-decoder framework. Pix2pix [12] maps an image to another domain. Pix2pix is fully supervised and requires pairs of matching

samples from the two domains. The method trains a GAN to distinguish between a "real" pair (source image, matching target image) and "fake"(source image, generated image) pairs. As mentioned, in many applications, the collection of these samples is both unpractical and hard to do accurately.

Style transfer In these methods [9], [14], [34], new images are synthesized by minimizing the content loss with respect to one input sample and the style loss with respect to one or more input samples. The content loss is typically the encoding of the image by a network training for an image categorization task, similar to our work. The style loss compares the statistics of the activations in various layers of the neural network. While style transfer was initially obtained by a slow optimization process [9], the emphasis was recently put on feed-forward methods [14], [34].

There are many links between style transfer and our work: both are unsupervised and generate a sample under f constancy, given an input sample. However, our work is much more general in its scope and does not rely on a predefined family of perceptual losses. Our domain transfer method can be used in order to perform style transfer, but not the other way around, since the style transfer methods cannot capture semantics. Another key difference is that the current style transfer methods are aimed at replicating the style of one or several images, while our work considers a distribution in the target space. In many applications, there is an abundance of unlabeled data in the target domain, which can be modeled accurately in an unsupervised manner. Finally, the Tied Output Synthesis problem that we solve differs, because the image we generate must adhere to specific constraints.

Given the impressive results of style transfer work, in particular for face images, one might get the false impression that emoji are just a different style of drawing faces. By way of analogy, this claim is similar to stating that a Siamese cat is a Labrador Retriever in a different style. Emoji differ from facial photographs in both content and style. Style transfer can create visually appealing face images. However, the properties of the target domain are compromised.

Similarly, the work that has been done to automatically generate sketches from images, e.g., [17], [35], [39], does not apply to our methods, since it typically trains in a supervised manner that requires correspondences between sketches and photographs. The literature of face sketches also does not produce a parameter vector in a semantic configuration space.

Unsupervised image translation A recent line of work [15], [38], [41], applies cycle-constraint to map images between two domains from an unpaired dataset. This is done by learning two generative functions $G : \mathcal{X} \to \mathcal{Y}$ and $F : \mathcal{Y} \to \mathcal{X}$ to minimize a three term loss: (a) GAN loss term for function G, (b) GAN loss term for function F and (c) a cycle constancy loss requiring $F(G(x)) \approx x$ and $G(F(y)) \approx y$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$. The methods above also employ GANs on input to generate images, instead of noise. These methods differ from our work, since they do not constrain the learned mapping with a perceptual function f. As a result, the mapping can be viewed as style-transfer e.g. applying the texture of the source domain on the target domain. For example, cats are not dogs in another style. However, the mapping in [41] generates cats with a dog texture. Our methods are able to utilize the additional supervision given by the pretrained function f, as is demonstrated in our experiments, which compare the results obtained with CycleGAN [41] and our Domain Transfer Network.

Distances between distributions In unsupervised learning, where one cannot match between an input sample and its output, many methods rely on measuring distances between distributions. Specifically, GANs were recently shown [8] to implement the theoretical notion of discrepancies.

Definition 1 (Discrepancy distance)**.** *Let $\mathcal{C}$ be a class of functions from A to B and let $\ell : B \times B \to \mathbb{R}_+$ be a loss function over B. The discrepancy distance* $\mathrm{disc}_{\mathcal{C}}$ *between two distributions D_1 and D_2 over A is defined as* $\mathrm{disc}_{\mathcal{C}}(D_1, D_2) = \sup_{c_1, c_2 \in \mathcal{C}} \left| R_{D_1}[c_1, c_2] - R_{D_2}[c_1, c_2] \right|$*, where* $R_D[c_1, c_2] = \mathbb{E}_{x \sim D}\left[\ell(c_1(x), c_2(x))\right]$.

Unsupervised Domain Adaptation The work done by [1], [3], [24] addresses the following problem: given a labeled training set in $\mathcal{X}_s \times \mathcal{Y}$, for some target space $\mathcal{Y}$, and an unlabeled set of samples from domain $\mathcal{X}_t$, learn a function $h : \mathcal{X}_t \to Y$.

The most successful recent work employs a GAN inspired solution, which uses a technique called gradient reversal that was introduced by Ganin et al. [8]. In this method, a learned discriminator is employed in order to learn an intermediate feature representation that is invariant to the source domain. Classification done on top of this representation is expected to be less sensitive to the shift in the domains.

Like our work, Unsupervised Domain Adaptation is a domain shift problem. Our work differs, since it focuses on generating aligned samples in the target domain, rather than labeling it. In Sec. 3, we thoroughly discuss the relation between our work and domain adaptation.

3 Problem Formulation

Problems involving domain shift receive an increasing amount of attention, since the field of machine learning moves its focus away from the vanilla supervised learning scenarios to new combinations of supervised, unsupervised and transfer learning. In this section, we formulate the two computational problems that we pose: (i) the free form "Cross Domain Transfer" problem, and (2) the parameter-based "Tied Output Synthesis" problem. In this section, these problems are put into a theoretical context. In the following sections, we redefine the problems as concrete deep learning problems. In order to maximize clarity, this section is kept as independent as possible from the following ones, and the reader may choose to skip the derivations and go directly to the architecture, as presented in Sec. 4 and Sec. 5.

3.1 Related Problem: Unsupervised Domain Adaptation

In the **unsupervised domain adaptation** problem [1], [3], [24], the algorithm trains a hypothesis on a source domain and the hypothesis is tested on a different target domain. The algorithm is aided with a labeled dataset of the source domain and an unlabeled dataset of the target domain. The conventional approach to dealing with this problem is to learn a feature map that (i) enables accurate classification in the source domain and (ii) captures meaningful invariant relationships between the source and target domains.

Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ be the output space. The source domain is a distribution D_S over $\mathcal{X}$ along with a function $y_S : \mathcal{X} \to \mathcal{Y}$. Similarly, the target domain is specified by (D_T, y_T). Given some loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ the goal is to fit a hypothesis h from some hypothesis space $\mathcal{H}$, which minimizes the *Target Generalization Risk*, $R_{D_T}[h, y_T]$. Where a *Generalization Risk* is defined as $R_D[h_1, h_2] = \mathbb{E}_{x \sim D}\left[\ell(h_1(x), h_2(x))\right]$.

	Input $\mathcal{X}$	Output $\mathcal{Y}$
1^{st}	$\{x_i \sim D_T\}$	
2^{nd}	$\{x_j \sim D_S\}$	$\{y_S(x_j)\}$

(a)

	Input $\mathcal{X}$	Output $\mathcal{Y}$
1^{st}	$\{x_i \sim D_1\}$	
2^{nd}		$\{y(x_j)\|x_j \sim D_2\}$

(b)

	Input $\mathcal{X}$	Out. $\mathcal{Y}_1$	Out. $\mathcal{Y}_2$
1^{st}	$\{x_i \sim D_1\}$		
2^{nd}		$e(c_j)$	$\{c_j \sim D_{\mathcal{Y}_2}\}$

(c)

Fig. 3. The domain shift configurations discussed in Sec. 3. (a) The unsupervised domain adaptation problem. The algorithm minimizes the risk in a target domain using training samples $\{(x_j \sim D_S, y_S(x_j))\}_{j=1}^m$ and $\{x_i \sim D_T\}_{i=1}^n$. (b) The unsupervised domain transfer problem. In this case, the algorithm learns a function G and is being tested on D_1. The algorithm is aided with two datasets: $\{x_i \sim D_1\}_{i=1}^m$ and $\{y(x_j) \sim D_2^y\}_{j=1}^n$. For example, in the facial emoji application, D_1 is the distribution of facial photos and D_2 is the (unseen) distribution of faces from which the observed emoji were generated. (c) The tied output synthesis problem, in which we are give a set of samples from one input domain $\{x_i \sim D_1\}$, and matching samples from two tied output domains: $\{(e(c_j), c_j)|c_j \sim D_{\mathcal{Y}_2}\}$.

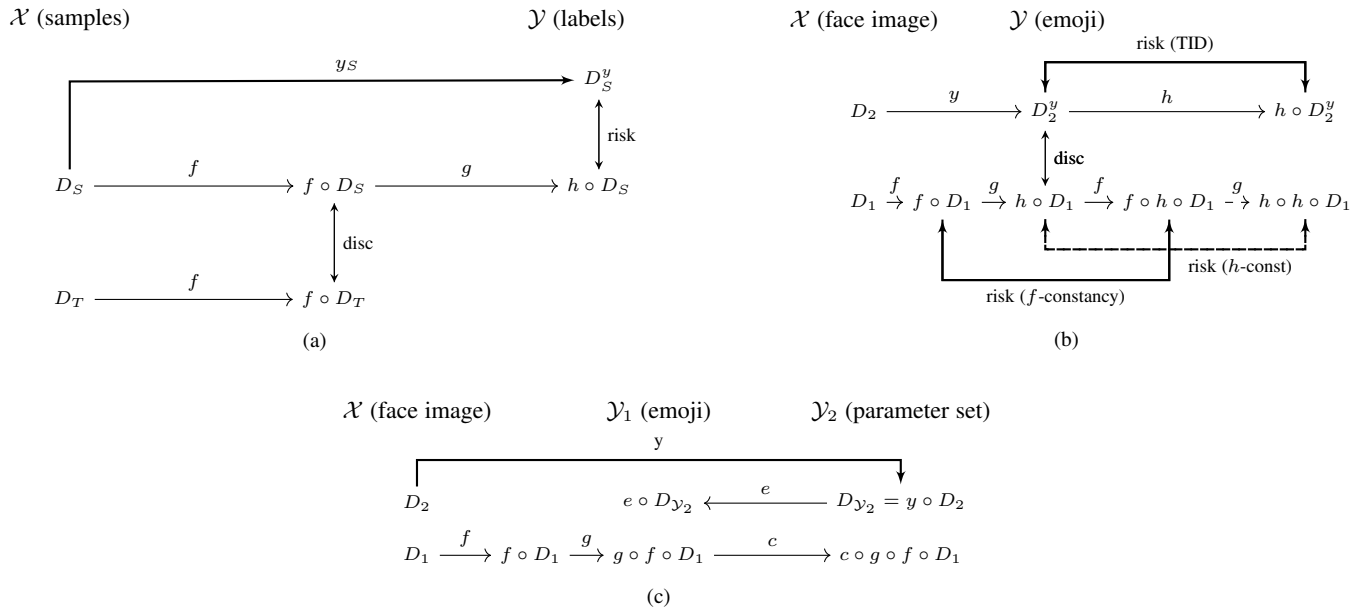


Fig. 4. Illustrations of the domain shift scenarios depicted in Sec. 3. (a) Unsupervised domain adaptation. Each node contains a distribution. The horizontal edges denote the mappings between the distributions and the learned function is $h = g \circ f$. The vertical edges denote the discrepancy between the the two distributions $f \circ D_S$ and $f \circ D_T$ and the risk between y and h on D_S. (b) Domain Transfer Network (DTN). The learned function is $h = g \circ f$. The horizontal two-sided edges denote the TID and f-constancy risks that are used by the algorithm. The vertical two-sided edge stands for the discrepancy between $D_2^y = y \circ D_2$ and $h \circ D_1$. The dashed edges stand for the h-constancy risk that is required only in Thm. 1. (c) Tied Output Synthesis. The unknown function y is learned by the approximation $h = c \circ g \circ f$. f and e are given. D_1 is the distribution of input images at test time. During training, we observe tied mappings $(y(x), e(y(x)))$ for unknown samples $x \sim D_2$ as well as unlabeled samples from the other distribution D_1. The risks that are shared with DTN are omitted for clarity. Figure credit: (a) and (b) are borrowed from [7]

The distributions D_S, D_T and the target function $y_T : \mathcal{X} \to \mathcal{Y}$ are unknown to the learning algorithm. Instead, the learning algorithm relies on a training set of labeled samples $\{(x, y_S(x))\}$, where x is sampled from D_S, as well as on an unlabeled training set of samples $x \sim D_T$, see Fig. 3(a).

In the common solutions to this problem, following [23], each hypothesis $h \in H$ is decomposed into a feature map f and a classifier g i.e. $h = g \circ f$. The function f represents inputs from $x \in \mathcal{X}$ as feature maps, so that the discrepancy distance (Def. 1) between $f \circ D_S$ and $f \circ D_T$ is minimal. The classifier, g, maps $f(\mathcal{X})$ to $\mathcal{Y}$. This is depicted in Fig. 4(a).

3.2 Problem I: Cross Domain Transfer

In the **cross domain transfer** problem, the task is to learn a function that maps samples from the input domain $\mathcal{X}$ to the output domain $\mathcal{Y}$. In our work, we show a GAN based solution able to convincingly transform face images into caricatures from a specific domain.

The training data available to the learning algorithm in the cross domain transfer problem is illustrated in Fig. 3(b) and Fig. 4(b). The problem consists of two distributions, D_1 and D_2, and a target function, y. The algorithm has access to the following two unsupervised datasets: $\{x_i \sim D_1\}_{i=1}^m$ and $\{y(x_j)|x_j \sim D_2\}_{j=1}^n$. The goal is to fit a function $h = g \circ f \in \mathcal{H}$ that optimizes $\inf_{h \in \mathcal{H}} R_{D_1}[h, y]$.

It is assumed that: (i) f is a fixed pre-trained feature map and, therefore, $\mathcal{H} = \{g \circ f | g \in \mathcal{H}_2\}$ for some hypothesis class $\mathcal{H}_2$; and (ii) y is idempotent, i.e, $y \circ y \equiv y$. For example, in our methods, f is the DeepFace representation [33] and y maps face images to emoji caricatures. In addition, applying y on an emoji gives the same emoji. Note that in the setting of cross domain transfer, D_1 and D_2 are the source and target distributions respectively. However, the loss $R_{D_1}[h, y]$ is measured over D_1, while in domain adaptation, it is measured over the target distribution.

Recently [7], the cross domain transfer problem was analyzed using the theoretical term of discrepancy. Denoting, for example, $y \circ D$ to be the distribution of the y mappings of samples $x \sim D$, then the following bound is obtained.

Theorem 1 (Domain transfer [7])**.** *If ℓ satisfies the triangle*

inequality[2] *and* $\mathcal{H}_2$ *(the hypothesis class of* g*) is a universal Lipschitz hypothesis class*[3]*, then for all* $h = g \circ f \in \mathcal{H}$*,*

$$\begin{aligned} R_{D_1}[h, y] \leq & R_{y \circ D_2}[h, \mathrm{Id}] + R_{D_1}[f \circ h, f] \\ & + \mathrm{disc}_{\mathcal{H}}(y \circ D_2, h \circ D_1) + \lambda \end{aligned} \tag{1}$$

Here, $\lambda = \min_{h \in \mathcal{H}} \{R_{y \circ D_2}[h, \mathrm{Id}] + R_{D_1}[h, y]\}$ *and* $h^* = g^* \circ f$ *is the corresponding minimizer.*

The theorem bounds the risk $R_{D_1}[h, y]$, i.e., the expected loss (using ℓ) between the mappings by the ground truth function y and the mapping by the learned function h for samples $x \sim \mathcal{D}_1$. Lastly, the λ factor captures the complexity of the hypothesis class $\mathcal{H}$, which depends on the chosen architecture of the neural network that instantiates g. A similar factor in the generalization bound of the unsupervised domain adaptation problem is presented in [1].

Comparing to Unsupervised Domain Adaptation We note that one can solve the cross domain transfer problem using domain adaptation and vice versa. In both cases, the solution is indirect. Solving domain adaptation using domain transfer can be done via the following steps: (i) First, one would learn a feature map function f using the samples $\{(x_j, y_S(x_j)) | x_j \sim D_S\}$, (ii) The pre-trained f will be used to apply the domain transfer algorithm in order to obtain a mapping from D_S to D_T (The function trained this way would be more accurate on samples $x \sim D_S$ than on $x \sim D_T$. This asymmetry is shared with all experiments done in this work) and (iii) Training samples from D_S could then be transferred to D_T and used to learn an adapted classifier that is suitable for the target domain.

In the other direction (solving domain transfer using a domain adaptation algorithm), given the function f, one can invert f by generating training samples $\{(f(x), x) | x \sim D_2\}$ and learn the function $h = f^{-1}$ from $f \circ D_2$ to D_2. Domain adaptation can then be used in order to learn mapping g from $f \circ D_1$ to D_2, thus achieving domain transfer. Based on the work by [40], we expect that h, even in the target domain of emoji, will be hard to learn, making this solution hypothetical at this point.

3.3 Problem II: Tied Output Synthesis

The second problem studied in this paper, is a third flavor of domain shift, which can be seen as a mix of the two problems: the unsupervised domain adaptation and the cross domain transfer problem. Similar to the unsupervised domain transfer problem, we are given a set of supervised labeled samples. The samples c_j are drawn i.i.d from the distribution $D_{\mathcal{Y}_2}$ in the space $\mathcal{Y}_2$ and are given together with their mappings $e(c_j) \in \mathcal{Y}_1$. In addition, and similar to the cross domain transfer problem, we are given samples $x_i \in \mathcal{X}$ drawn i.i.d from another distribution D_1. The goal is to learn a mapping $y : \mathcal{X} \to \mathcal{Y}_2$ that satisfies the following condition $y \circ e \circ y = y$. The hypothesis class contains functions h of the form $c \circ g \circ f$ for some known f for $g \in \mathcal{H}_2$ and for $c \in \mathcal{H}_3$. f is a pre-learned function that maps the input sample in $\mathcal{X}$ to some feature space, g maps from this feature space to the space $\mathcal{Y}_1$, and c maps from this space to the space of parameters $\mathcal{Y}_2$, see Fig. 3(c) and Fig. 4(c).

Our approach assumes that e is prelearned from the matching samples $(c_j, e(c_j))$. However, c is learned together with g. This makes sense, since while e is a feedforward transformation from a set of parameters to an output, c requires the conversion of an input of the form $g(f(x))$ where $x \sim D_1$, which is different from the image of e for inputs in $\mathcal{Y}_2$. The theorem below describes our solution. It assumes that $D_{\mathcal{Y}_2} = y \circ D_2$, for some distribution D_2 of samples in $\mathcal{X}$.

Theorem 2 (Tied output bound)**.** *If* ℓ *satisfies the triangle inequality and* $\mathcal{H}_2$ *is a universal Lipschitz hypothesis class with respect to* ℓ*, then for all* $h = c \circ g \circ f \in \mathcal{H}$*,*

$$\begin{aligned} R_{D_1}[e \circ h, e \circ y] \leq & R_{D_1}[e \circ h, g \circ f] + R_{e \circ y \circ D_2}[g \circ f, \mathrm{Id}] \\ & + R_{D_1}[f \circ g \circ f, f] \\ & + \mathrm{disc}_{\mathcal{H}}(e \circ y \circ D_2, g \circ f \circ D_1) + \lambda, \end{aligned} \tag{2}$$

where $\lambda = \min_{g \in \mathcal{H}_2} \{R_{e \circ y \circ D_2}[g \circ f, \mathrm{Id}] + R_{D_1}[g \circ f, e \circ y]\}$ *and* g^* *is the corresponding minimizer.*

Proof. By the triangle inequality, we obtain:
$R_{D_1}[e \circ h, e \circ y] \leq R_{D_1}[e \circ h, g \circ f] + R_{D_1}[g \circ f, e \circ y]$.
Applying Thm. 1 completes the proof:

$$\begin{aligned} R_{D_1}[g \circ f, e \circ y] \leq & R_{e \circ y \circ D_2}[g \circ f, \mathrm{Id}] + R_{D_1}[f \circ g \circ f, f] \\ & + \mathrm{disc}_{\mathcal{H}}(e \circ y \circ D_2, g \circ f \circ D_1) + \lambda \quad \square \end{aligned}$$

Thm. 2 presents a recursive connection between the tied output synthesis problem and the cross domain transfer problem. This relation can be generalized for tying even more outputs to even more complex relations among parts of the training data. The importance of having a generalization bound to guide our solution stems from the plausibility of many other terms, such as $R_{e \circ y \circ D_2}[e \circ h, g \circ f]$ or $R_{D_1}[f \circ g \circ f, f \circ e \circ h]$.

Comparing to Unsupervised Cross Domain Transfer The tied output problem is a specific case of cross domain transfer with $\mathcal{Y}$ of the latter being $\mathcal{Y}_1 \times \mathcal{Y}_2$ of the former. However, this view makes no use of the network e. Comparing Thm. 1 and Thm. 2, there is an additional term in the second bound: $R_{D_1}[e \circ h, g \circ f]$. It expresses the expected loss (over samples from D_1) when comparing the result of applying the full cycle of encoding by f, generating an image by g, estimating the parameters in the space $\mathcal{Y}_2$ using c, and synthesizing the image that corresponds to these parameters using e, to the result of applying the subprocess that includes only f and g.

Comparing to Unsupervised Domain Adaptation Consider the domain $\mathcal{X} \cup \mathcal{Y}_1$ and learn the function e^{-1} from this domain to $\mathcal{Y}_2$, using the samples $\{(e(c_j), c_j) | c_j \sim D_2\}$, adapted to $x_i \sim D_1$. This is a domain adaptation problem with $D_S = e \circ D_2$ and $D_T = D_1$. Our experiments show that applying this reduction leads to suboptimal results. This is expected, since this approach does not make use of the prelearned feature map f. This feature map is not to be confused with the feature network learned in [8], which we denote by p. The latter is meant to eliminate the differences between $p \circ D_S$ and $p \circ D_T$. However, the prelearned f leads to easily distinguishable $f \circ D_S$ and $f \circ D_T$.

The unsupervised domain adaptation and the TOS problem become more similar, if one identifies p with the conditional function that applies $g \circ f$ to samples from $\mathcal{X}$ and the identity to samples from $\mathcal{Y}_1$. In this case, the label predictor of [8] is identified with our c and the discrepancy terms (i.e., the GANs) are applied to the same pairs of distributions. However, the two solutions would still

2. For all $y_1, y_2, y_3 \in \mathcal{Y}$ it holds that $\ell(y_1, y_3) \leq \ell(y_1, y_2) + \ell(y_2, y_3)$. This holds for the absolute loss, and can be relaxed to the square loss, where it holds up to a multiplicative factor of 3.

3. A function $c \in \mathcal{C}$ is Lipschitz with respect to ℓ, if there is a constant $L > 0$ such that: $\forall a_1, a_2 \in A : \ell(c(a_1), c(a_2)) \leq L \cdot \ell(a_1, a_2)$. A hypothesis class $\mathcal{C}$ is universal Lipschitz with respect to ℓ, if all functions $c \in \mathcal{C}$ are Lipschitz with some universal constant $L > 0$. This holds, for example, for neural networks with leaky ReLU activations and weight matrices of bounded norms, under the squared or absolute loss.

differ, since (i) our solution minimizes $R_{D_1}[e \circ h, g \circ f]$, while in unsupervised domain adaptation, the analog term is minimized over $D_S = e \circ D_2$ and (ii) the additional non-discrepancy terms would not have analogs in the domain adaptation bounds.

4 Networks for Free-form Generation

We next reformulate the problems as neural network challenges. For clarity, this formulation is purposefully written to be independent of the mathematical presentation above.

In the Domain Transfer problem (Sec. 3.2), we study the task of projecting an image in one domain to an image in another domain. Given a domain, $\mathcal{X}$, and a function f, we would like to learn a generative function G, such that f is invariant under G, i.e., $f \circ G = f$. In network form, f maps an image to a fixed size 1-dimensional representation. f architecture is not constrained by our method. Exact architectures used in our experiments are provided in Sec. 6.1. The training data is unsupervised and consists of a set of samples from the source domain $\mathcal{X}$ and a second set from the target domain $\mathcal{Y}$.

4.1 A baseline formulation

Given a set $\mathbf{s}$ of unlabeled samples in a source domain $\mathcal{X}$ sampled i.i.d according to some distribution $\mathcal{D}_1$, a set of samples in the target domain $\mathbf{t} \subset \mathcal{Y}$ sampled i.i.d from distribution $\mathcal{D}_2$, a function f from the domain $\mathcal{X}$ and a weight α, we wish to learn a function $G : \mathcal{X} \to \mathcal{Y}$ that minimizes the combined risk $R = R_{\text{GAN}} + \alpha R_{\text{CONST}}$, which is comprised of

$$R_{\text{GAN}} = \max_d \mathbb{E}_{x\sim\mathcal{D}_1} \log[1-d(G(x))] + \mathbb{E}_{x\sim\mathcal{D}_2} \log[d(x)], \quad (3)$$

where d is a binary classification function from $\mathcal{Y}$, $d(x)$ the probability of the class 1 it assigns for a sample $x \in \mathcal{Y}$, and

$$R_{\text{CONST}} = \mathbb{E}_{x\sim\mathcal{D}_1} \|f(x) - f(G(x))\|^2 \quad (4)$$

The first term is the adversarial risk, which requires that for every discriminative function d, the samples from the target domain would be indistinguishable from the samples generated by G for samples in the source domain.

The second term is the f-constancy term, which requires that f is invariant under G. In practice, we have experimented with multiple distance metrics, including Mean Squared Error (MSE) and cosine distance, as well as other variants including metric learning losses (hinge) and triplet losses. The performance is mostly unchanged, and we report results using the simplest MSE solution.

Similar to other GAN formulations, one can minimize the loss associated with the risk R over G, while maximizing it over d, where G and d are deep neural networks, and the expectations in R are replaced by summations over the corresponding training sets:

$$L_{\text{GAN}} = \max_d \sum_{x\in\mathbf{s}} \log[1 - d(G(x))] + \sum_{x\in\mathbf{t}} \log[d(x)], \quad (5)$$

$$L_{\text{CONST}} = \sum_{x\in\mathbf{s}} \|f(x) - f(G(x))\|^2 \quad (6)$$

In our experiments, we show that this baseline does not produce desirable results.

4.2 The domain transfer network

Domain transfer network (DTN) is a more elaborate architecture that contains two high level modifications used to solve the cross domain transfer problem. First, we employ $f(x)$ as the baseline representation to the function G. Second, we consider, during training, the generated samples $G(x)$ for $x \in \mathbf{t}$.

The first change is stated as $G = g \circ f$, for some learned function g. By applying this, we focus the learning effort of G on the aspects that are most relevant to R_{CONST}. In addition, in most applications, f is not as accurate on $\mathcal{Y}$ as it is on $\mathcal{X}$. The composed function, which is trained on samples from both $\mathcal{X}$ and $\mathcal{Y}$, adds layers on top of f, which adapt it.

The second change alters the form of L_{GAN}, making it multi-class, instead of binary. It also introduces a new term L_{TID} that requires G to be the identity matrix on samples from $\mathcal{Y}_1$. Taken together and written in terms of training loss, we now have two losses L_D and $L_G = L_{\text{GANG}} + \alpha L_{\text{CONST}} + \beta L_{\text{TID}} + \gamma L_{\text{TV}}$, for some weights α, β, γ, where

$$L_{\text{D}} = -\sum_{x\in\mathbf{s}} \log d_1(g(f(x))) - \sum_{x\in\mathbf{t}} \log d_2(g(f(x))) - \sum_{x\in\mathbf{t}} \log d_3(x) \quad (7)$$

$$L_{\text{GANG}} = -\sum_{x\in\mathbf{s}} \log d_3(g(f(x))) - \sum_{x\in\mathbf{t}} \log d_3(g(f(x))) \quad (8)$$

$$L_{\text{TID}} = \sum_{x\in\mathbf{t}} \|x - G(x)\|^2 \quad (9)$$

and where d is a ternary classification function from the domain $\mathcal{Y}_1$ to $1, 2, 3$, and $d_i(x)$ is the probability it assigns to class $i = 1, 2, 3$ for an input sample x. L_{CONST} is explained by Eq. 6. During optimization, L_G is minimized over g and L_D is minimized over d. See Fig. 5(b) for an illustration of our method.

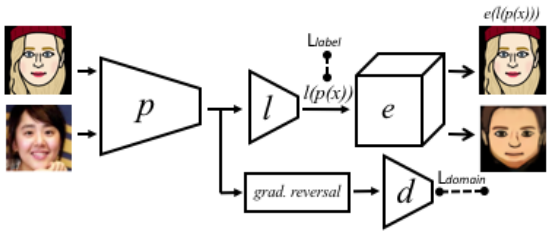
Eq. 7 and 8 make sure that the generated analogy, i.e., the output of G, is in the target space $\mathcal{Y}$. Since d is ternary and can, therefore, confuse classes in more than one way, this role, which is captured by Eq. 3 in the baseline formulation, is split into two. However, the two equations do not enforce any similarity between the source sample x and the generated $G(x)$. This is done by Eq. 6 and 9: Eq. 6 enforces f-constancy for $x \in \mathcal{X}$, while Eq. 9 enforces that for samples $x \in \mathcal{Y}$, which are already in the target space, G is the identity mapping. The latter is a desirable behavior, e.g., for the cartooning task, given an input emoji, one would like it to remain constant under the mapping of G. It can also be seen as an autoencoder type of loss, applied only to samples from $\mathcal{Y}$. The experiments reported in Sec. 6 evaluate the contributions of L_{CONST} and L_{TID} and reveal that at least one of these is required, and that when employing only one loss, L_{CONST} leads to a better performance than L_{TID}.

The last loss, L_{TV} is an anisotropic total variation loss [22], [31], which is added in order to slightly smooth the resulting image. The loss is defined on the generated image $z = [z_{ij}] = G(x)$ as

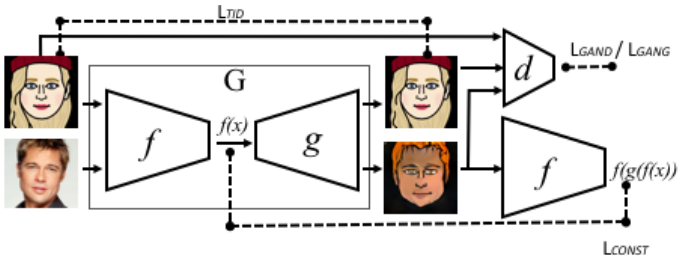
$$L_{TV}(z) = \sum_{i,j} \left((z_{i,j+1} - z_{ij})^2 + (z_{i+1,j} - z_{ij})^2 \right)^{\frac{B}{2}}, \quad (10)$$

where we employ $B = 1$.

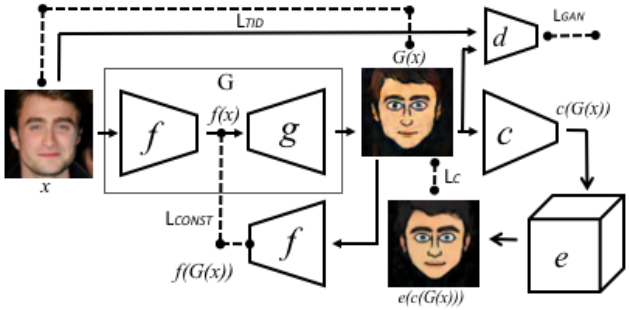
The Domain Transfer Network matches Thm. 1. The first term in the R.H.S of Thm. 1, $R_{y\circ D_2}[h, \text{Id}]$, is the L_{TID} part of the DTN loss, which, for the cartooning task, states that emoji caricatures are mapped to themselves. The second term $R_{D_1}[f \circ h, f]$ corresponds to the L_{CONST} term of DTN, which states that the under



(a)



(b)



(c)

Fig. 5. (a) The training constraints for unsupervised domain adaptation applied to the problem of recovering the avatar parameters given an input image. The learned functions are p, l and d. *grad. revesal* stands for the gradient reversal operation during back propagation. The mapping e is assumed to be known a-priori and is used only during inference time to generate an emoji from the configurations. (b) The training constraints of the Domain Transfer Network method. The learned functions are d and $G = g \circ f$, for a given f. The dashed lines denote loss terms. (c) The training constraints of the Tied Output Synthesis Network. The learned functions are c, d, and $G = g \circ f$, for a given f. The mapping e is assumed to be known a-priori. The dashed lines denote loss terms.

f-constancy the input face image and the resulting caricature are similar. The theorem shows that this constancy does not need to be assumed and is a result of the idempotency of y and the structure of h. The third term $\text{disc}_{\mathcal{H}}(y \circ D_2, h \circ D_1)$ is the GAN element of the method, which compares generated caricatures ($h \circ D_1$) to the training dataset of the unlabeled emoji ($y \circ D_2$).

5 Networks for Parameterized Generation

The Tied Output Synthesis problem presented in Sec. 3.3 extends the Domain Transfer problem of Sec. 3.2. Given a domain, $\mathcal{X}$, a mapping e and a function f, we would like to learn a generative function G, such that f is invariant under G, and that for all samples $x \in \mathcal{X}$, there exists a configuration $u \in \mathcal{Y}_2$ such that $G(x) = e(u)$. In comparison to the Domain Transfer problem, the target domain of TOS ($\mathcal{Y}_1$) is constrained to be the image of a mapping e.

A solution to the Domain Transfer problem cannot satisfy the additional constraint imposed by Tied Output Synthesis, since presenting it with a training set $\mathbf{t}$ of samples generated by e is not a strong enough constraint. Furthermore, the real-world avataring applications require the recovery of the configuration u itself, which allows the synthesis of novel samples using an extended engine e^* that generates new poses, expressions in the case of face images, etc. From the application point of view, the tied output synthesis problem tackled here is considerably more applicable than the domain transfer problem.

5.1 Applying Unsupervised Domain Adaptation

The Domain Adaptation architecture of [8] can be used to solve the TOS challenge. The algorithm is given a training set $\mathbf{s} \subset \mathcal{X}$, and a paired training set $(t_1, t_2) \subset \mathcal{Y}_1 \times \mathcal{Y}_2$.

Learning is done by training three networks p,l and d so that: (a) $e^{-1}(t_1) = t_2$ where $e^{-1} = l \circ p$ and (b) network d is unable to determine the input domain of the sample $x \in \mathcal{X} \cup \mathcal{Y}_1$ based on the representation $p(x)$. In loss terms, condition (a) is met by p and l minimizing:

$$L_{\text{label}} = \sum_{x \in t_1} \|l(p(x)) - t_2\|^2 \tag{11}$$

and condition (b) is met by d minimizing and p maximizing:

$$L_{\text{domain}} = \sum_{x \in s} \log[1 - d(p(x))] + \sum_{x \in t} \log[d(p(x))], \tag{12}$$

maximization is achieved by taking the reverse of the gradient (multiply by -1) during back propagation of L_{domain}. Fig. 5(a) depicts the full architecture. In our experiments, we show the result of this approach is not identifiable as our proposed solution.

5.2 The interplay between the trained networks

Before presenting our solution to the TOS problem, we present a general description of the framework we use in the learning process. The constraint of TOS, generating images in a tied domain of image and configuration, requires the learned networks to share a mutual signal during training to tie both domains. Thus, the framework is presented in contrast to the adversarial framework of GANs.

In a general view of GANs, assume a loss function $\ell(G, d, x)$, for some function d that receives inputs in the domain $\mathcal{Y}_1$. G, which maps an input x to entities in $\mathcal{Y}_1$, minimizes the risk of Eq. 3, which can be rewritten as: $R_{\text{GAN}} = \max_d - \mathbb{E}_x \ell(G, d, x)$. This optimization is successful, if **for every** function d, the expectation of $\ell(G, d, x)$ is small for the learned G. It is done by maximizing this expectation with respect to d, and minimizing it with respect to G. The two learned networks d and G provide a training signal to each other.

Two networks can also provide a mutual signal, by collaborating on a shared task. Consider the case in which G and a second function c work hand-in-hand in order to minimize the expectation of some other loss $\ell(G, c, x)$. In this case, G "relies" on c and minimizes the following expression:

$$R_c = \min_c \mathbb{E}_x \ell(G, c, x). \tag{13}$$

Algorithm 1 The TOS training algorithm.

Given the function $e : \mathcal{Y}_2 \to \mathcal{Y}_1$, an embedding function f, and $\mathbf{S} \subset \mathcal{X}$, $\mathbf{T} \subset \mathcal{Y}_1$ training sets.
Initialize networks c, g and d
while iter $<$ numiters **do**
  Sample mini-batches $\mathbf{s} \subset \mathbf{S}$, $\mathbf{t} \subset \mathbf{T}$
  Compute feed-forward $d(t)$, $d(g(f(s)))$
  Update d by minimizing L_{GAN} ▷ Eq. 5
  Update g by maximizing L_{GAN} ▷ Eq. 5
  Update g by minimizing L_{CONST} ▷ Eq. 6
  Update g by minimizing L_{TID} ▷ Eq. 9
  Update g by minimizing L_{TV}
  Compute $e(c(z))$ by feed-forwarding $z := g(f(s))$
  Update c and g by minimizing L_{c} ▷ Eq. 15

This optimization succeeds, if **there exists** a function c for which, post-learning, the expectation $\mathbb{E}_x\, \ell(G, c, x)$ is small.

In the problem of tied output synthesis, the function e maps entities u in some configuration space $\mathcal{Y}_2$ to the target space $\mathcal{Y}_1$. c maps samples from $\mathcal{Y}_1$ to the configuration space, essentially inverting e. The suitable loss is:

$$\ell_e(G, c, x) = \|G(x) - e(c(G(x)))\|^2. \tag{14}$$

For such a problem, the optimal c is given by $c^*(z) = \operatorname{argmin}_u \|z - e(u)\|^2$. This implicit function is intractable to compute, and c is learned instead as a deep neural network.

5.3 The TOS Network

The solution of the Tied Output Synthesis problem is given a mapping e, perceptual function f, a training set $\mathbf{s} \subset \mathcal{X}$, and a training set $\mathbf{t} \subset \mathcal{Y}_1$. Similar to DTN, we define G to be composed out of f and a second function g that maps from the output space of f to $\mathcal{Y}_1$, i.e., $G = g \circ f$. The e compliance term (L_c of Eq. 13 using ℓ_e of Eq. 14) becomes:

$$L_c = \sum_{x \in \mathbf{s}} \|g(f(x)) - e(c(g(f(x))))\|^2 \tag{15}$$

The compliance term is added to the constraints of DTN (Equations (6) to (10)) and results in two training losses, d minimizes L_D, and both g and c minimize

$$L_G = L_c + \alpha L_{\text{GAN}} + \beta L_{\text{CONST}} + \gamma L_{\text{TID}} + \delta L_{\text{TV}} \tag{16}$$

for some non-negative weights $\alpha, \beta, \gamma, \delta$. The method is illustrated in Fig. 5(c) and laid out in Alg. 1. Exact architectures are provided in Sec. 6.1.

In the context of Thm. 2, the term L_c corresponds to the risk term $R_{D_1}[e \circ h, g \circ f]$ in the theorem and compares samples transformed by the mapping $g \circ f$ to the mapping of the same samples to a configuration in $\mathcal{Y}_2$ using $c \circ g \circ f$ and then to $\mathcal{Y}_1$ using e. The term L_{TID} corresponds to the risk $R_{e \circ y \circ D_2}[g \circ f, \text{Id}]$, which is the expected loss over the distribution from which $\mathbf{t}$ is sampled, when comparing the samples in this training set to the result of mapping these by $g \circ f$. The discrepancy term $\text{disc}_{\mathcal{H}}(e \circ y \circ D_2, g \circ f \circ D_1)$ matches the L_{GAN} term, which, as explained above, measures a distance between two distributions, in this case, $e \circ y \circ D_2$, which is the distribution from which the training set $\mathbf{t}$ is taken, and the distribution of mappings by $g \circ f$ of the samples $\mathbf{s}$ which are drawn from D_1.

Fig. 6. Domain transfer from SVHN domain to MNIST domain. Input in odd columns; output in even columns.

6 Experiments

In the following section, we evaluate both the free-form Domain Transfer Network and the parameter-based Tied Output Synthesis methods. The Domain Transfer Network is evaluated in the digits domain, where we transfer images from the Street View House Number (SVHN) dataset [26] to the domain of the MNIST dataset [18]. We perform an ablation study to investigate the importance of Equations (6) to (10). Additionally, we show that DTN can be used to perform domain adaptation.

The Tied Output Synthesis is evaluated on a toy problem of inverting a polygon synthesizing engine. We use this toy-case to illustrate the method and the contribution of incorporating e. Finally, we evaluate both systems on the task of avatar generation from a photograph for two different CG engines.

6.1 Architecture

The networks used in our experiments are inspired by DCGAN [29]. For our DTN experiments on the Digits datasets, network f is a modification of the DCGAN discriminator to output 128D representation, instead of real/fake probability.

Network e is based on DCGAN's generator architecture, except for mapping a configuration vector, instead of noise, to an RGB image. Since the online emoji rendering engine is additive in nature and contains a finite number of options for each facial feature (nose, eyes, hair, etc.), a network with enough capacity mimics it without difficulty and in our experiments, we found the DCGAN inspired architecture to be sufficient. The same architecture of network e was used for both the simple Polygons dataset and for the Face Emoji dataset. Since polygons are simpler, it is likely that a simpler architecture will suffice. However, we preferred using the same architecture.

Network c is based on DCGAN's discriminator and it predicts the configuration vector (instead of real/fake probability), given an RGB image. The detailed architectures are given in Tab. 1.

6.2 Digits

For working with digits, we employ the training split of SVHN, which contains 531,131 images for two purposes: learning the function f and as an unsupervised training set $\mathbf{s}$ for the domain transfer method. The evaluation is done on the test split of SVHN,

TABLE 1
Architectures of networks used in experiments of Sec. 6. The digits dataset was used to evaluate the DTN method and, therefore, does not employ networks e and c. The synthetic Polygons dataset did not use f-constancy therefore does not include network f.

	Digits	Face Emoji	Polygons
f	Four 4×4 convolutional layers with 64-128-256-128 filters, followed by max pooling. All layers employ ReLU activation.	DeepFace [33]	—
d	Four 4×4 convolutional layers with 64-128-256-1 filters. All layers, except last one, employ batch normalization, and a leaky ReLU with leakiness coefficient of 0.2. All the layers use a stride of 2 and padding of 1, except the last one, which does not use stride or padding.	Six 4×4 convolutional layers with 64-128-256-512-512-3 filters. All layers, except last one, employ batch normalization, and a leaky ReLU with leakiness coefficient of 0.2. All the layers use a stride of 2 and padding of 1, except the last one, which does not use stride or padding.	Same as FaceEmoji
g	Four 4×4 upscaling convolutional layers with 64-128-256-1 filters. All layers employ batch normalization, and ReLU activation, except the last layer, which employs Tanh activation. All the layers use a stride of 2 and padding of 1, except the first one, which does not use stride or padding.	Nine convolutional layers. All layers employ batch normalization, and ReLU activation except last one which employs Tanh activation. The odd layers perform upscaling 4×4 convolutions with 512-256-128-64-3 filters. The even layers perform 1×1 convolutions [19]. The odd layers use a stride of 2 and padding of 1, except the first one, which does not use stride or padding.	Same as FaceEmoji without the 1×1 convolutional layers i.e., five convolutional layers with 512-256-128-64-3 filters
c	—	Five convolutional layers with 64-128-256-512-813 filters. All layers employ batch normalization, and ReLU activation, except the last layer, which employs Tanh activation.	Same as FaceEmoji
e	—	Five 4×4 upscaling convolutional layers with 512-256-128-64-3 filters. All layers employ batch normalization and ReLU activation. The last layer employs Tanh activation. All the layers use a stride of 2 and padding of 1, except the first one, which does not use stride or padding.	Same as FaceEmoji

comprised of 26,032 images. The error on the test split is 4.95%. Even though this accuracy is far from the best reported results, it seems to be sufficient for the purpose of domain transfer. Within the DTN, f maps a 32×32 RGB image to the activations of the last convolutional layer of size $128 \times 1 \times 1$ (post a 4×4 max pooling and before the ReLU). In order to apply f on MNIST images, we replicate the grayscale image three times, obtaining a monochromatic RGB image.

The set $\mathbf{t}$ contains the test set of the MNIST dataset. For supporting quantitative evaluation, we have trained a classifier on the train set of the MNIST dataset, consisting of the same architecture as f. The accuracy of this classifier on the test set approaches perfect performance at 99.4% accuracy, and is, therefore, trustworthy as an evaluation metric. In comparison, the network f achieves 76.08% accuracy on $\mathbf{t}$.

Network g, maps SVHN-trained f's 128D representations to 32×32 grayscale images. In the digit experiments, the results were obtained with the tradeoff hyperparamemters $\alpha = \beta = 15$. We did not observe a need to add a smoothness term and the weight of L_{TV} was set to $\gamma = 0$.

Despite not being very accurate on both domains (and also considerably worse than the SVHN state of the art), we were able to achieve visually appealing domain transfer, as shown in Fig. 6.

In order to provide a quantitative evaluation, we have employed the MNIST network on the set of samples $G(\mathbf{s}_{TEST}) = \{G(x)|x \in \mathbf{s}_{TEST}\}$, using the true SVHN labels of the test set. We first compare to the baseline method of Sec. 4.1, where the generative function, which works directly with samples in $\mathcal{X}$, is composed out of a few additional layers at the bottom of G. The results, shown in Tab. 2, demonstrate that DTN has a clear advantage over the baseline method. In addition, the contribution of each one of the terms in the loss function is shown in the table. The regularization term L_{TID} seems less crucial than the constancy term. However, at least one of them is required in order to obtain good performance. The GAN constraints are also important. Finally, the inclusion of f within the generator function G has a dramatic influence on the results.

TABLE 2
Accuracy of the MNIST classifier on the sampled transferred by our DTN method from SHVN to MNIST.

Method	Accuracy
Baseline method (Sec. 4.1)	13.71%
CycleGAN [41]	26.1%
DistanceGAN [2]	26.8%
DTN	90.66%
DTN w/0 L_{TID}	88.40%
DTN w/0 L_{CONST}	74.55%
DTN G does not contain f	36.90%
DTN w/0 L_{D} and L_{GANG}	34.70%
DTN w/0 L_{CONST} & L_{TID}	5.28%
Original SHVN image	40.06%

Also shown are the results of experiments which evaluate the performance of DistanceGAN [2] and CycleGAN [41] on the task of domain transfer, by transferring the test set of SVHN to MNSIT space and using the MNIST network as the classifier. The results demonstrate the advantage of DTN over mapping methods, which do not employ the perceptual function f during the learning.

As explained in Sec. 3.2, domain transfer can be used in order to perform unsupervised domain adaptation. For this purpose, we transformed the set $\mathbf{s}$ to the MNIST domain (as above), and using the true labels of $\mathbf{s}$, employed a simple nearest neighbor classifier there. The choice of classifier was to emphasize the simplicity of the approach; However, the constraints of the unsupervised domain transfer problem would be respected for any classifier trained on $G(\mathbf{s})$. The results of this experiment are reported in Tab. 3, which shows a clear advantage over the method of

TABLE 3
Domain adaptation from SVHN to MNIST

Method	Accuracy
SA [6]	59.32%
DANN [8]	73.85%
DTN on SVHN transferring the train split $\mathbf{s}$	84.44%
DTN on SVHN transferring the test split	79.72%

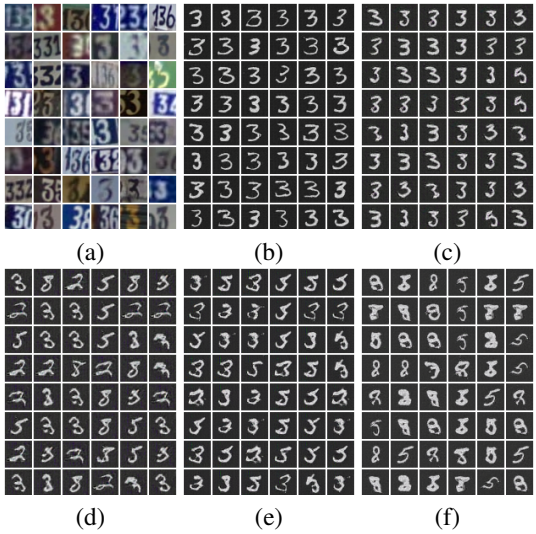


Fig. 7. A random subset of the digit '3' from SVHN, transferred to MNIST. (a) The input images. (b) The results of our DTN. In all plots, the cases keep their respective locations, and are sorted by the probability of '3', as inferred by the MNIST classifier on the results of our DTN. (c) The obtained results, in which the digit 3 was not shown as part of the set $\mathbf{s}$ unlabeled samples from SVNH. (d) The obtained results, in which the digit 3 was not shown as part of the set $\mathbf{t}$ of unlabeled samples in MNIST. (e) The digit 3 was not shown in both $\mathbf{s}$ and $\mathbf{t}$. (f) The digit 3 was not shown in $\mathbf{s}$, $\mathbf{t}$, and during the training of f.

DANN [8]. This is true both when transferring the samples of the set $\mathbf{s}$ and when transferring the test set of SVHN, which is much smaller and was not seen during the training of the DTN.

6.2.1 Unseen digits

Another set of experiments was performed in order to study the ability of the domain transfer network to overcome the omission of a class of samples. This type of ablation can occur in the source or the target domain, or during the training of f and can help us understand the importance of each of these inputs. The results are shown visually in Fig. 7, and qualitatively in Tab. 4, based on the accuracy of the MNIST classifier only on the transferred samples from the test set of SVHN that belong to class '3'.

It is evident that not including the class in the source domain is much less detrimental than eliminating it from the target domain. This is the desirable behavior: never seeing any '3'-like shapes in $\mathbf{t}$, the generator should not generate such samples. The results are better when not observing '3' in both $\mathbf{s}, \mathbf{t}$ than when not seeing it only in $\mathbf{t}$, since in the latter case, G learns to map source samples of '3' to target images of other classes.

6.3 Polygons

TOS is initially evaluated in a context that is independent of f constancy. Given a set of images $\mathbf{t} \in \mathcal{Y}_1$, and a mapping e from some vector space to $\mathcal{Y}_1$, learn a mapping c and a generative function G that creates e-compliant random images in $\mathcal{Y}_1$ (Eq. 14).

We create binary 64×64 images of regular polygons by sampling uniformly three parameters: the number of vertices (3-6), the radius of the enclosing circle (15-30), and a rotation angle in the range $[-10, 10]$. Some polygons are shown in Fig. 8(a). 10,000 training images were created and used in order to train a CNN e that maps the three parameters to the output, with very little loss (MSE of 0.1). The MNIST dataset does not contain configuration u for each sample. Therefore, it is not suitable for assessing the benefits of e supervision.

A training set $\mathbf{t}$ of a similar size is collected by sampling in the same way. As a baseline method, we employ DCGAN [29], in which the input x is a random vector in $[-1, 1]^{100}$. The results are shown in Fig. 8(b). While the generated images are similar to the class of generated polygons, they are not from this class and contain visible artifacts, such as curved edges.

A TOS network is then trained by minimizing Eq. 14 with the additional GAN constraints. The optimization minimizes $L_c + \alpha L_{\text{GAN}}$, for $\alpha = 1$ (L_{CONST} and L_{TID} are irrelevant to this experiment), and with the input distribution D_1 of random vectors sampled uniformly in the $[-1, 1]$ hypercube in 100D. The results, as depicted in Fig. 8(c), show that TOS, which enjoys the additional supervision of e, produces results that better fit the polygon class. See Sec. 6.1 for details on the architectures.

TABLE 4
Comparison of recognition accuracy of the digit 3 as generated in MNIST

Method	Accuracy of '3'
DTN	94.67%
'3' was not shown in $\mathbf{s}$	93.33%
'3' was not shown in $\mathbf{t}$	40.13%
'3' was not shown in both $\mathbf{s}$ or $\mathbf{t}$	60.02%
'3' was not shown in $\mathbf{s}$, $\mathbf{t}$, and during the training of f	4.52 %

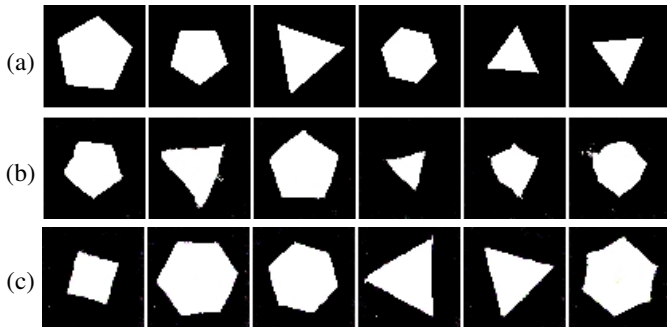


Fig. 8. Toy problem. (a) Polygon images with three random parameters: number of vertices, radius of enclosing circle and rotation. (b) GAN generated images mimicking the class of polygon images. (c) $G(x)$ images created by TOS. The TOS is able to benefit from the synthesis engine e and produces images that are noticeably more compliant than the GAN.

6.4 Face Emoji

The proposed TOS method is evaluated for the task of generating specification-compliant emoji. In this task, we transfer an "in-the-wild" facial photograph to a set of parameters that defines an emoji. As the unlabeled training data of face images (domain $\mathcal{X}$), we use a set $\mathbf{s}$ of one million random images without identity

information. The face images were cropped and aligned into 152×152 RGB images, as done in [33]. The set $\mathbf{t}$ consists of assorted facial avatars (*emoji*) created by an online service (bitmoji.com). The emoji images were processed by an automatic process that detects, based on a set of heuristics, the center of the irises and the tip of the nose. Based on these coordinates, the emoji were centered and scaled into 152×152 RGB images.

The emoji engine of the online service is mostly additive. In order to train the TOS, we mimic it and have created a neural network e that maps properties such as gender, length of hair, shape of eyes, etc. into an output image. Network e maps emoji parameterization into the matching 64×64 RGB emoji. The parameterization is given as binary vectors in $\mathbb{R}^{813}$ for emojis; Avatar parameterization is in $\mathbb{R}^{354}$. While there are dependencies among the various dimensions (an emoji cannot have two hairstyles at once), the binary representation is chosen for its simplicity and generality. e is trained in a fully supervised way, using pairs of matching parameterization vectors and images in a supervised manner.

The networks composing TOS are described in Sec. 6.1. Network e is pretrained to support the TOS methods. As function f, we employ the representation layer of the DeepFace network [33]. This representation is 256-dimensional and was trained on a labeled set of four million images that does not intersect the set $\mathbf{s}$. Network g maps DeepFace's 256-dimensional representation [33] into 64×64 RGB emoji images.

TOS is compared to the DTN method, which does not enforce the generated emoji to be specification-compliant. The DTN employs the same architecture for networks f, d, g, albeit different choice of hyperparameters $\alpha = 100$, $\beta = 1$, $\gamma = 0.05$ which were selected via validation. Network d takes 152×152 RGB images (either natural or scaled-up emoji) and outputs log-probabilities predicting if the image is fake or real. Finally, network c of the TOS method maps a 64×64 emoji to a parameterization vector with values in range $[-1, 1]$. We set $\alpha = 0.01$, $\beta = 100$, $\gamma = 1$, $\delta = 0.0005$ as the tradeoff hyperparameters, after eyeballing the results of the first epoch of a very limited set of experiments.

For evaluation purposes only, we employ a benchmark which contains manually created emoji of 118 random images from the CelebA dataset [21]. The benchmark was created by a team of professional annotators, who used the web service that creates the emoji images. Fig. 9 shows side by side samples of the original image, the human generated emoji, the emoji generated by the generator function of DTN, and the emoji generated by both the generator $G = g \circ f$ and the compound generator $e \circ c \circ G$ of our TOS method. As can be seen, the DTN emoji tend to be more informative, albeit less restrictive than the ones created manually. TOS respects the configuration space and creates emoji that are similar to the ones created by the human annotators, but which tend to carry more identity information.

6.4.1 Identifiability

In order to evaluate the identifiability of the resulting emoji, we have collected a second example for each identity in the set of 118 CelebA images and a set $\mathbf{s}'$ of 100,000 random face images (unsupervised, without identity), which were not included in $\mathbf{s}$. The VGG face CNN descriptor [28] is then used in order to perform retrieval as follows. For each image x in the manually annotated set, a gallery $\mathbf{s}' \cup x'$ is created, where x' is the other image of the person in x. Retrieval is then performed using VGG faces and either the manually created emoji, $G(x)$, or $e(c(G(x)))$ as the probe.

TABLE 5
A comparison of median rank for retrieval out of a set of 100,001 face images for either manually created emoji, or emoji and VR avatars created by DTN or TOS. The results are shown for the "raw" $G(x)$, as well as for the configuration compliant $e(..(x))$. Since DTN does not produce a configuration-compliant emoji, we obtain the results for the $e(..(x))$ column, by applying to its output a pretrained network $\bar{c}$ that maps emoji to configurations. Also shown are DANN results obtained when training such a mapping $\bar{c}$ that is adapted to the samples in $\mathbf{s}$.

Method	Emoji		Avatars	
	$\mathbf{g}(\mathbf{f}(\mathbf{x}))$	$\mathbf{e}(..(\mathbf{x}))$	$\mathbf{g}(\mathbf{f}(\mathbf{x}))$	$\mathbf{e}(..(\mathbf{x}))$
Manual	NA	16,311	NA	NA
DANN [8]	NA	59,625	NA	52,435
DTN	16	18,079	195	38,805
TOS	30	3,519	758	11,153
TOS fixed $\bar{c}$	26	14,990	253	43,160

In these experiments, the VGG face network is used in order to avoid a bias that might be caused by using f both for training the DTN and the TOS methods and for evaluation. The results are reported in Tab. 5(left). As can be seen, the $G(x)$ emoji generated by DTN are extremely discriminative and obtain a median rank of 16 in cross-domain identification out of 10^5 distractors. However, DTNs are not compatible with any configuration vector. In order to demonstrate this, we trained a network $\bar{c}$ that maps emoji images to configurations. When applied to the emoji generated by DTN and transforming the results, using e, back to an emoji, the obtained images are less identifiable than the emoji created manually (Tab. 5, under $e(..(x))$). By comparison, the median rank of the emoji created by the configuration vector $c(G(x))$ of TOS is much better than the result obtained by the human annotators. As expected, DTN has more identifiable results than TOS, when considering the output of $g(f(x))$ directly, since TOS has additional terms and the role of L_{CONST} in TOS is reduced.

The need to train c and G jointly, as is done in the TOS framework, is also verified in a second experiment, in which we pretrained network c to be the network $\bar{c}$. The results of rendering the configuration vector were also not as good as those obtained by the unmodified TOS framework. As expected, querying by $G(x)$ directly, produces results that are between DTN and TOS.

It should be noted that using the pretrained $\bar{c}$ directly on input faces, leads to fixed configurations (modes), since $\bar{c}$ was trained to map from $\mathcal{Y}_1$ and not from $\mathcal{X}$. This is also true when the prediction is based on f mappings of the input and when training a mapping from $\mathcal{X}$ to $\mathcal{Y}_2$ under the f distance on the resulting avatar.

6.4.2 Domain Adaptation

This situation calls for the use of unsupervised domain adaptation (Sec. 3) to learn a mapping from $\mathcal{X}$ to $\mathcal{Y}_2$ by adapting a mapping from $\mathcal{Y}_1$. Despite some effort, applying the domain adaptation method of [8] did not result in satisfactory results (Tab. 5 and Fig. 10), since the method does not preserve the image identity.

In the domain adaptation method, network p extracts 2048-dimensional feature vectors from 64×64 RGB images. It resembles the structure of network c - with 4 convolution layers. Each convolution is with 64-128-256-512 filters respectively. The last convolutional layer employs a stride of 1, instead of 2 and does not use batch-normalized or leaky ReLU. Finally, the network output is flattened to a 1-dimensional feature vector.

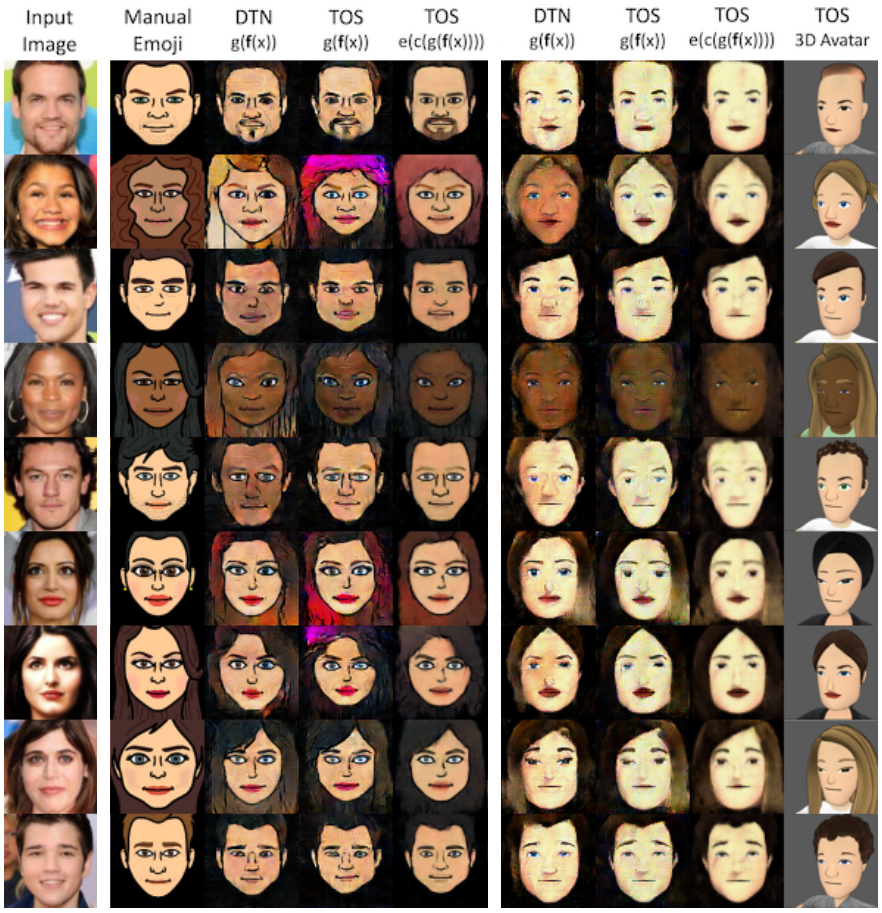


Fig. 9. Shown, side by side, are (a) sample images from the CelebA dataset. (b) emoji, from left to right: the images created manually using a web interface (for evaluation only), the result of DTN, and the two results of our TOS: $G(x)$ and then $e(c(G(x)))$. (c) VR avatar results: DTN, the two TOS results, and a 3D rendering of the resulting configuration file. See Tab. 5 for retrieval performance.

The label prediction network l accepts as input, feature vectors generated by p and outputs emoji parameterization vectors matching the input image. It consists of three fully connected layers. Each hidden layer is followed by batch-normalization and leaky ReLU activation. The last layer is followed by Tanh activation. The hidden layers contain 1024 and 512 units respectively.

The discriminator d predicts the input image domain given its feature vector. It consists of two fully connected layers with 512 hidden units. The hidden layer is followed by batch normalization and leaky ReLU activations. It is preceded by a gradient reversal layer to ensure that the feature distributions of both domains are similar. The last layer is followed by Sigmoid activation.

6.4.3 Human rating

Finally, we asked a group of 20 volunteers to select the better emoji, given a photo from celebA and two matching emoji: one created by the expert annotators and one created by TOS ($e \circ c \circ G$). The raters were told that they are presented with the results of two algorithms for automatically generating emoji and were requested to pick their favorable emoji for each image. The images were presented printed out, in random order, and the raters were given an unlimited amount of time. In 39.53% of the answers, the TOS emoji was selected. This is remarkable, considering that in a good portion of the celebA emoji, the TOS created very dark emoji in an unfitting manner (since f is invariant to illumination and since the configuration has many more dark skin tones than lighter ones). TOS, therefore, not only provides more identifiable emoji, but is also very close to be on par with professional annotators. It is important to note that we did not compare to DTN in this rating, since DTN does not create a configuration vector, which is needed for avatar applications (Fig 2).

6.4.4 Multiple Images Per Person

We evaluate the results obtained per person and not just per image on the Facescrub dataset [27]. For each person q, we considered the set of their images X_q, and selected the emoji that was most similar to their source image, i.e., the one for which:

$$\underset{x \in X_q}{\operatorname{argmin}} ||f(x) - f(e(c(G(x))))|| \tag{17}$$

The qualitative results are appealing and are shown in Fig. 11. The general problem of mapping a set $X \subset \mathcal{X}$ to a single output in $\mathcal{Y}$ is left for future work.

6.5 VR Avatars

We next apply the proposed TOS method to a commercial avatar generator engine, see Fig. 9(c). We sample random parameterizations and automatically align their frontally-rendered avatars into 64×64 RGB images to form the training set $\mathbf{t}$. We then train a CNN e to mimic this engine and generate such images

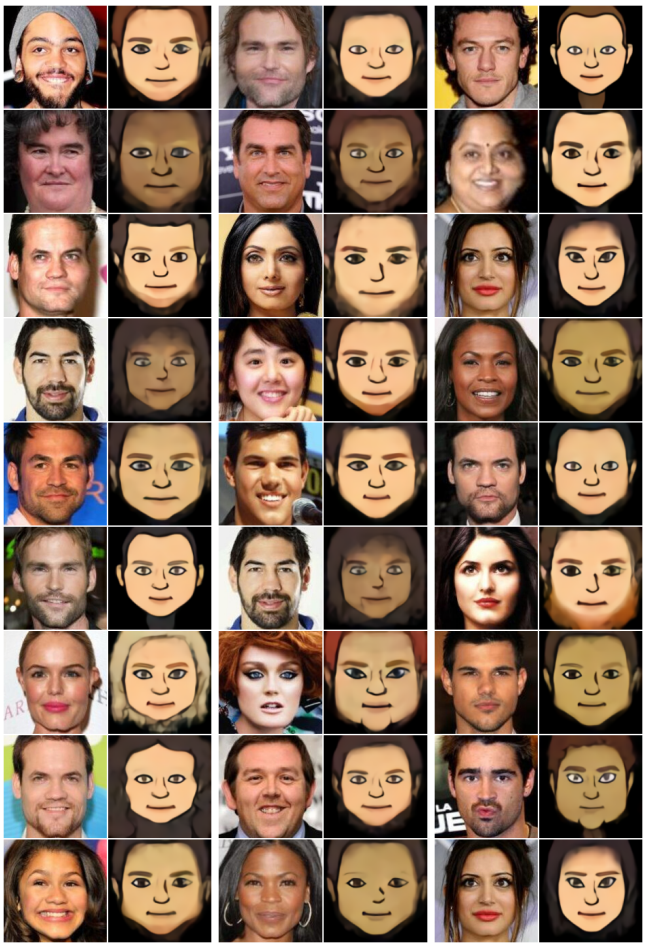


Fig. 10. Shown, side by side are sample images from the CelebA dataset and the results obtained by the DANN domain adaptation method [8]. These results are not competitive.

Fig. 11. Multi-image results obtained by the TOS method for a sample of individuals from the Facescrub dataset. Shown, side by side, are the image used to create the TOS and the DTN emoji, the DTN emoji, and the TOS emoji, obtained by $e \circ c \circ g \circ f$. The image that represents a person maximizes, out of all images for this person, f-constancy for the TOS method.

given their parameterization. Using the same architectures and configurations as in Sec. 6.4, including the same training set $\mathbf{s}$, we train g and c to map natural facial photographs to their engine-compliant set of parameters. We also repeat the same identification experiment and report median rankings of the analog experiments, see Tab. 5(right). The 3D avatar engine is by design, not as detailed as the 2D emoji one, with elements such as facial hair still missing and less part shapes available. In addition, the avatar model style is more generic and focused on real time puppeteering and not on cartooning. Therefore, the overall numbers are lower for all methods, as expected. TOS seems to be the only method that is able to produce identifiable configurations, while the other methods lead to ranking that is close to random.

7 Conclusions

With the advent of better computer graphics engines and the plethora of available models, and the ability of neural networks to compare cross-domain entities, the missing element for bridging between computer vision and computer graphics is the ability to link image data to a suitable computer graphics model. The presented DTN method created analogies without explicit supervision. Highly identifiable emoji were generated; However, emoji applications call for parametrized characters, which can then be transformed by artists to other views and new expressions. DTN does not output these configurations and, as we show, the emoji created by DTN cannot be converted to a configuration. The TOS network is able to generate identifiable emoji that are coupled with a valid configuration vector. This is done by jointly training two networks, g and c, in order to achieve a common goal.

While TOS was presented in a way that requires the rendering function e to be differentiable, working with black-box renderers using gradient estimation techniques is a common practice, e.g., in Reinforcement Learning, and the simple REINFORCE [36] method can be readily used.

Our experiments have focused on the specific emoji domain, in which solutions are both lacking and in high demand. However, in a broader view, TOS addresses one of the most fundamental paradigms in computational vision and, more generally, in perception – the one of analysis by synthesis. In this paradigm, there are three major subroutines: First, given an input, a hypothesis is conceived. Second, a representation of the hypothesis is created. The domain of this representation would typically be more abstract than the domain of the input. Third, the representation is compared to the input. The machinery required for the second and third steps (image synthesis and distance learning, respectively) has been the focus of much study. Our method is unique in its ability to provide the missing hypothesis generation tool with no additional supervision. More concretely: the input is $x \in \mathcal{X}$, the generated hypothesis is $c(G(x))$, the representation of the hypothesis is

$e(c(g(x))) \in \mathcal{Y}_1$, and the comparator employs f and some metric.

The TOS network method is both unsupervised (no correspondences between the hypotheses and the input space are required) and able to bridge the semantic gap between $\mathcal{X}$ and $\mathcal{Y}_1$ via f. Furthermore, for iterative paradigms, which employ analysis by synthesis loops, the error signal of the verification step $\|g(x) - e(c(g(x)))\|$ can be used to update the hypothesis during runtime, or, to train a second network, which given $G(x)$ and the error signal of the previous iteration, updates the hypothesis. Thus, TOS fully addresses all aspects of the analysis by synthesis framework.

REFERENCES

[1] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2).

[2] S. Benaim and L. Wolf. One-Sided Unsupervised Domain Mapping. In *NIPS*, 2017.

[3] K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. *J. Mach. Learn. Res.*, 9:1757–1774, June 2008.

[4] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*. 2016.

[5] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015.

[6] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, 2013.

[7] T. Galanti and L. Wolf. A theory of output-side unsupervised domain adaptation. *arXiv preprint arXiv:1703.01606*, 2017.

[8] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030, Jan. 2016.

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *NIPS*. 2014.

[11] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra. DRAW: a recurrent neural network for image generation. In *ICML*, 2015.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[13] D. Jimenez Rezende, S. M. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. In *NIPS*. 2016.

[14] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[15] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. In *ICML*, 2017.

[16] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*. 2015.

[17] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics*, 2013.

[18] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[19] M. Lin, Q. Chen, and S. Yan. Network In Network. In *ICLR*, 2014.

[20] M.-Y. Liu and O. Tuzel. Coupled Generative Adversarial Networks. In *NIPS*. 2016.

[21] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[22] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

[23] Y. Mansour. Learning and domain adaptation. In *Algorithmic Learning Theory, 20th International Conference, ALT*, pages 4–6, 2009.

[24] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.

[25] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[26] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[27] H. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *ICIP*, 2014.

[28] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[29] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[30] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[31] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. In *International Conference of the Center for Nonlinear Studies on Experimental Mathematics : Computational Issues in Nonlinear Science*, pages 259–268, 1992.

[32] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017.

[33] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[34] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

[35] N. Wang, D. Tao, X. Gao, X. Li, and J. Li. Transductive face sketch-photo synthesis. *IEEE transactions on neural networks and learning systems*, 24(9):1364–1376, 2013.

[36] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

[37] L. Wolf, Y. Taigman, and A. Polyak. Unsupervised creation of parameterized avatars. In *ICCV*, 2017.

[38] Z. Yi, H. Zhang, P. Tan, and M. Gong. DualGAN: Unsupervised dual learning for image-to-image translation. In *ICCV*, 2017.

[39] Y. Zhang, N. Wang, S. Zhang, J. Li, and X. Gao. Fast face sketch synthesis via kd-tree search. In *ECCV*, 2016.

[40] A. Zhmoginov and M. Sandler. Inverting face embeddings with convolutional neural networks. *arXiv preprint arXiv:1606.04189*, 2016.

[41] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

Adam Polyak is a PhD student under the supervision of Prof, L. Wolf from Tel-Aviv University and a Research Engineer in the Facebook AI Research (FAIR) Group. He received the bachelors degree in computer science and mathematics from Bar-Ilan University as part of the program for mathematically talented youth, and the masters (Hons.) (magna cum laude) degree under the guidance of Prof. L. Wolf.

Yaniv Taigman graduated from Tel-Aviv University with a Masters in Computer Science. While pursuing his PhD research, he co-founded Face.com where he held the position of CTO. When Face.com was acquired by Facebook in 2012, he joined the office in Menlo Park to lead research and engineering projects. During this time, he worked on efficient methods for face recognition (DeepFace project), and helped start the AI group. In 2016, he established a satellite FAIR team in Tel-Aviv.

Lior Wolf is a Research Scientist in the Facebook AI Research (FAIR) Group and a Full Professor at the School of Computer Science at Tel-Aviv University. Previously, he was a post-doctoral associate in Prof. Poggio's lab at MIT and he graduated from the Hebrew University, Jerusalem, where he worked under the supervision of Prof. Shashua. He is an ERC grantee and has received the ICCV 2001 Marr Prize honorable mention and the best paper awards at ECCV 2000, the post ICCV 2009 workshop on eHeritage, the pre-CVPR2013 workshop on action recognition and ICANN 2016.