UNSUPERVISED CROSS-DOMAIN IMAGE GENERATION

Yaniv Taigman, Adam Polyak & Lior Wolf
Facebook AI Research
Tel-Aviv, Israel
{yaniv,adampolyak,wolf}@fb.com

ABSTRACT

We study the problem of transferring a sample in one domain to an analog sample in another domain. Given two related domains, S and T, we would like to learn a generative function G that maps an input sample from S to the domain T, such that the output of a given representation function f, which accepts inputs in either domains, would remain unchanged. Other than f, the training data is unsupervised and consist of a set of samples from each domain, without any mapping between them. The Domain Transfer Network (DTN) we present employs a compound loss function that includes a multiclass GAN loss, an f preserving component, and a regularizing component that encourages G to map samples from T to themselves. We apply our method to visual domains including digits and face images and demonstrate its ability to generate convincing novel images of previously unseen entities, while preserving their identity.

1 INTRODUCTION

Humans excel in tasks that require making analogies between distinct domains, transferring elements from one domain to another, and using these capabilities in order to blend concepts that originated from multiple source domains. Our experience tells us that these remarkable capabilities are developed with very little, if any, supervision that is given in the form of explicit analogies.

Recent achievements replicate some of these capabilities to some degree: Generative Adversarial Networks (GANs) are able to convincingly generate novel samples that match that of a given training set; style transfer methods are able to alter the visual style of images; domain adaptation methods are able to generalize learned functions to new domains even without labeled samples in the target domain and transfer learning is now commonly used to import existing knowledge and to make learning much more efficient.

These capabilities, however, do not address the general analogy synthesis problem that we tackle in this work. Namely, given separated but otherwise unlabeled samples from domains S and T and a perceptual function f, learn a mapping $G : S \to T$ such that $f(x) \sim f(G(x))$.

In order to solve this problem, we make use of deep neural networks of a specific structure in which the function G 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 Generative Adversarial Network (GAN) term that encourages the creation of samples $G(x)$ that are indistinguishable from the training samples of the target domain, regardless of $x \in S$ or $x \in T$. The second loss term enforces that for every x in the source domain training set, $||f(x) - f(G(x))||$ is small. The third loss term is a regularizer that encourages G to be the identity mapping for all $x \in T$.

The type of problems we focus on in our experiments are visual, although our methods are not limited to visual or even to perceptual tasks. Typically, f would be a neural network representation that is taken as the activations of a network that was trained, e.g., by using the cross entropy loss, in order to classify or to capture identity.

As a main application challenge, we tackle the problem of emoji generation for a given facial image. Despite a growing interest in emoji and the hurdle of creating such personal emoji manually, no system has been proposed, to our knowledge, that can solve this problem. Our method is able to

produce face emoji that are visually appealing and capture much more of the facial characteristics than the emoji created by well-trained human annotators who use the conventional tools.

2 RELATED WORK

As far as we know, the *domain transfer* problem we formulate is novel despite being ecological (i.e., appearing naturally in the real-world), widely applicable, and related to cognitive reasoning (Fauconnier & Turner, 2003). In the discussion below, we survey recent GAN work, compare our work to the recent image synthesis work and make links to unsupervised domain adaptation.

GAN (Goodfellow et al., 2014) methods train a generator network G that synthesizes samples from a target distribution given noise vectors. 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.

While originally proposed for generating random samples, GANs can be used as a general tool to measure equivalence between distributions. Specifically, the optimization of D corresponds to taking the most discriminative D achievable, which in turn implies that the indistinguishability is true for every D. Formally, Ganin et al. (2016) linked the GAN loss to the H-divergence between two distributions of Ben-david et al. (2006).

The generative architecture that we employ is based on the successful architecture of Radford et al. (2015). There has recently been a growing concern about the uneven distribution of the samples generated by G – that they tend to cluster around a set of modes in the target domain (Salimans et al., 2016). In general, we do not observe such an effect in our results, due to the requirement to generate samples that satisfy specific f-constancy criteria.

A few contributions ("Conditional GANs") have employed GANs in order to generate samples from a specific class (Mirza & Osindero, 2014), or even based on a textual description (Reed et al., 2016). When performing such conditioning, one can distinguish between samples that were correctly generated but fail to match the conditional constraint and samples that were not correctly generated. This is modeled as a ternary discriminative function D (Reed et al., 2016; Brock et al., 2016).

The recent work by Dosovitskiy & Brox (2016), has shown promising results for learning to map embeddings to their pre-images, given input-target pairs. Like us, they 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. In contrast, 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.

Another class of very promising generative techniques that has recently gained traction is neural style transfer. In these methods, 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. We do not employ style losses in our method. While initially style transfer was obtained by a slow optimization process (Gatys et al., 2016), recently, the emphasis was put on feed-forward methods (Ulyanov et al., 2016; Johnson et al., 2016).

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 method can be used in order to perform style transfer, but not the other way around. 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 T, which can be modeled accurately in an unsupervised manner.

Given the impressive results of recent 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 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.

In the computer vision literature, work has been done to automatically generate sketches from images, see Kyprianidis et al. (2013) for a survey. These systems are able to emphasize image edges and facial features in a convincing way. However, unlike our method, they require matching pairs of samples, and were not shown to work across two distant domains as in our method. Due to the lack of supervised training data, we did not try to apply such methods to our problems. However, one can assume that if such methods were appropriate for emoji synthesis, automatic face emoji services would be available.

Unsupervised domain adaptation addresses the following problem: given a labeled training set in $S \times Y$, for some target space Y, and an unlabeled set of samples from domain T, learn a function $h : T \to Y$ (Chen et al., 2012; Ganin et al., 2016). One can solve the sample transfer problem (our problem) using domain adaptation and vice versa. In both cases, the solution is indirect. In order to solve domain adaptation using domain transfer, one would learn a function from S to Y and use it as the input method of the domain transfer algorithm in order to obtain a map from S to T[1]. The training samples could then be transferred to T and used to learn a classifier there.

In the other direction, given the function f, one can invert f in the domain T by generating training samples $(f(x), x)$ for $x \in T$ and learn from them a function h from $f(T) = \{f(x)|x \in T\}$ to T. Domain adaptation can then be used in order to map $f(S) = \{f(x)|x \in S\}$ to T, thus achieving domain transfer. Based on the work by Zhmoginov & Sandler (2016), we expect that h, even in the target domain of emoji, will be hard to learn, making this solution hypothetical at this point.

3 A BASELINE PROBLEM FORMULATION

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

$$R_{\text{GAN}} = \max_D \mathbb{E}_{x \sim \mathcal{D}_S} \log[1 - D(G(x))] + \mathbb{E}_{x \sim \mathcal{D}_T} \log[D(x)], \tag{1}$$

where D is a binary classification function from T, $D(x)$ the probability of the class 1 it assigns for a sample $x \in T$, and

$$R_{\text{CONST}} = \mathbb{E}_{x \sim \mathcal{D}_S} d(f(x), f(G(x))) \tag{2}$$

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. An adversarial risk is not the only option. An alternative term that does not employ GANs would directly compare the distribution $\mathcal{D}_T$ to the distribution of $G(x)$ where $x \sim \mathcal{D}_S$, e.g., by using KL-divergence.

The second term is the f-constancy term, which requires that f is invariant under G. In practice, we have experimented with multiple forms of d 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.

Similarly 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. However, this baseline solution, as we will show experimentally, does not produce desirable results.

[1]The function trained this way would be more accurate on S than on T. This asymmetry is shared with all experiments done in this work.

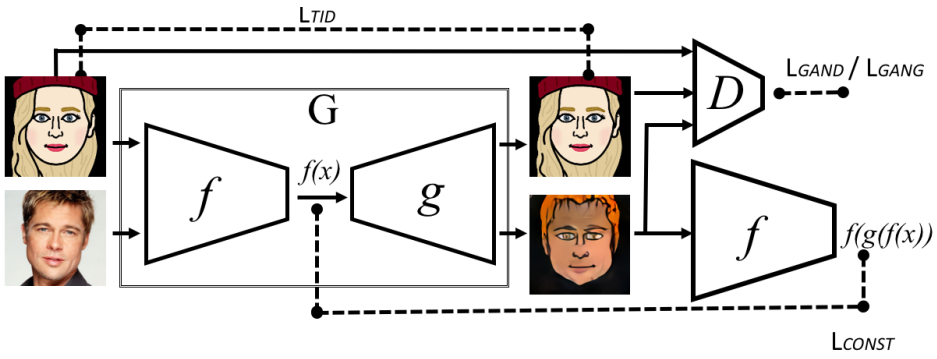


Figure 1: The Domain Transfer Network. Losses are drawn with dashed lines, input/output with solid lines. After training, the forward model G is used for the sample transfer.

4 THE DOMAIN TRANSFER NETWORK

We suggest to employ a more elaborate architecture that contains two high level modifications. 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 T as it on S. The composed function, which is trained on samples from both S and T, adds layers on top of f, which adapt it.

The second change alters the form of L_{GAN}, making it multiclass instead of binary. It also introduces a new term L_{TID} that requires G to be the identity matrix on samples from T. 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

$$L_{\text{D}} = -\sum_{x \in \mathbf{s}} \log D_1(g(f(x))) - \sum_{x \in \mathbf{t}} \log D_2(g(f(x))) - \sum_{x \in \mathbf{t}} \log D_3(x) \tag{3}$$

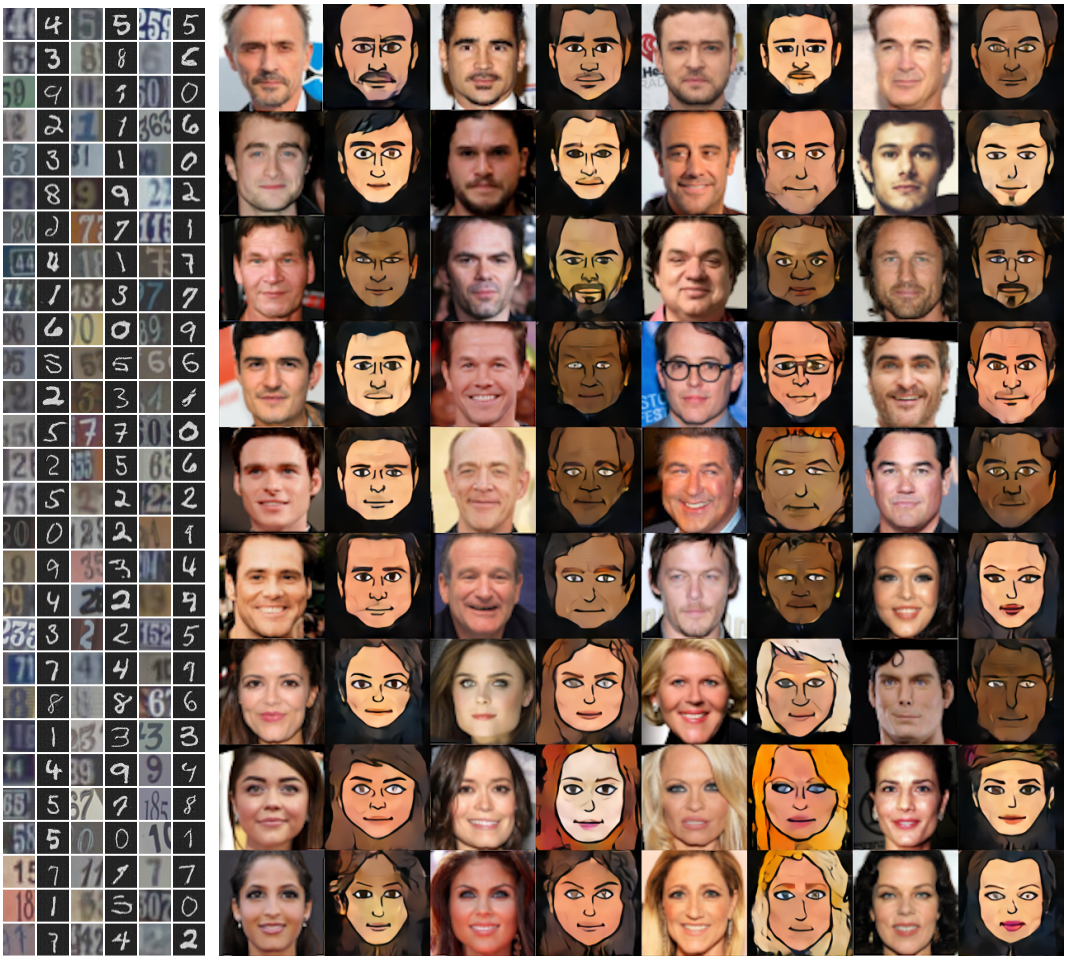
$$L_{\text{GANG}} = -\sum_{x \in \mathbf{s}} \log D_3(g(f(x))) - \sum_{x \in \mathbf{t}} \log D_3(g(f(x))) \tag{4}$$

$$L_{\text{CONST}} = \sum_{x \in \mathbf{s}} d(f(x), f(g(f(x)))) \tag{5}$$

$$L_{\text{TID}} = \sum_{x \in \mathbf{t}} d_2(x, G(x)) \tag{6}$$

and where D is a ternary classification function from the domain T to $1, 2, 3$, and $D_i(x)$ is the probability it assigns to class $i = 1, 2, 3$ for an input sample x, and d_2 is a distance function in T. During optimization, L_G is minimized over g and L_D is minimized over D. See Fig. 1 for an illustration of our method.

Eq. 3 and 4 make sure that the generated analogy, i.e., the output of G, is in the target space T. Since D is ternary and can therefore confuse classes in more than one way, this role, which is captured by Eq. 1 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. 5 and 6: Eq. 5 enforces f-constancy for $x \in S$, while Eq. 6 enforces that for samples $x \in T$, 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 T. The experiments reported in Sec. 5 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}.



(a) (b)

Figure 2: Domain transfer in two visual domains. Input in odd columns; output in even columns. (a) Transfer from SVHN to MNIST. (b) Transfer from face photos (Facescrub dataset) to emoji.

The last loss, L_{TV} is an anisotropic total variation loss (Rudin et al., 1992; Mahendran & Vedaldi, 2015), 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

$$L_{TV}(z) = \sum_{i,j} \left((z_{i,j+1} - z_{ij})^2 + (z_{i+1,j} - z_{ij})^2 \right)^{\frac{B}{2}}, \tag{7}$$

where we employ $B = 1$.

In our work, MSE is used for both d and d_2. We also experimented with replacing d_2, which, in visual domains, compares images, with a second GAN. No noticeable improvement was observed. Throughout the experiments, the adaptive learning rate method Adam by Kingma & Ba (2016) is used as the optimization algorithm.

5 Experiments

The Domain Transfer Network (DTN) is evaluated in two application domains: digits and face images. In the first domain, we transfer images from the Street View House Number (SVHN) dataset of Netzer et al. (2011) to the domain of the MNIST dataset by LeCun & Cortes (2010). In

Table 1: Accuracy of the MNIST classifier on the sampled transferred by our DTN method from SHVN to MNIST.

Method	Accuracy
Baseline method (Sec. 3)	13.71%
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%

Table 2: Domain adaptation from SVHN to MNIST

Method	Accuracy
SA Fernando et al. (2013)	59.32%
DANN Ganin et al. (2016)	73.85%
DTN on SVHN transferring the train split $\mathbf{s}$	84.44%
DTN on SVHN transferring the test split	79.72%

the face domain, we transfer a set of random and unlabeled face images to a space of emoji images. In both cases, the source and target domains differ considerably.

5.1 DIGITS: FROM SVHN TO MNIST

For working with digits, we employ the extra 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, comprised of 26,032 images. The architecture of f consists of four convolutional layers with $64, 128, 256, 128$ filters respectively, each followed by max pooling and ReLU non-linearity. The error on the test split is 4.95%. Even tough 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.

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, inspired by Radford et al. (2015), maps SVHN-trained f's 128D representations to 32×32 grayscale images. g employs four blocks of deconvolution, batch-normalization, and ReLU, with a hyperbolic tangent terminal. The architecture of D consists of four batch-normalized convolutional layers and employs ReLU. See Radford et al. (2015) for more details on the networks architecture. 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. 2(a). In order to evaluate the contribution of each of the method's components, 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. 3, where the generative function, which works directly with samples in S, is composed out of a few additional layers at the bottom of G. The results, shown in Tab. 1, 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.

As explained in Sec. 2, domain transfer can be used in order to perform unsupervised domain adaptation. For this purposes, 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

Table 3: Comparison of recognition accuracy of the digit 3 as generated in MNIST

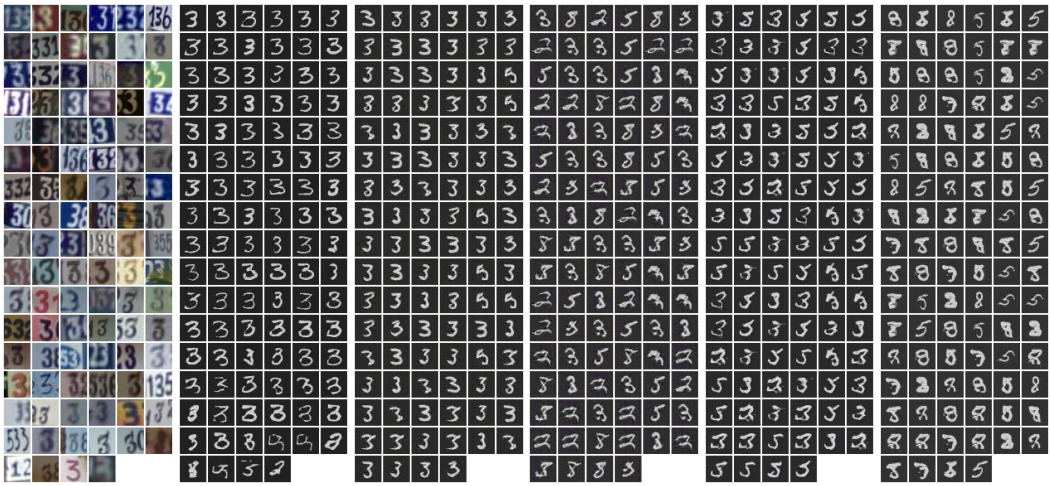
Method	Accuracy of '3'
DTN	94.67%
'3' was not shown in $\mathbf{s}$	93.33%
'3' was not shown in $\mathbf{t}$	40.13%
'3' was not shown in both $\mathbf{s}$ or $\mathbf{t}$	60.02%
'3' was not shown in $\mathbf{s}$, $\mathbf{t}$, and during the training of f	4.52 %

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. 2, which shows a clear advantage over the state of the art method of Ganin et al. (2016). 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.

5.1.1 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. 3, and qualitatively in Tab. 3, 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. 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.



(a) (b) (c) (d) (e) (f)

Figure 3: A random subset of the digit '3' from SVHN, transferred to MNIST. (a) The input images. (b) 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.

Table 4: Comparison of retrieval accuracy out of a set of 100,001 face images for either manually created emoji or the one created by the DTN method.

Measure	Manual	Emoji by DTN
Median rank	16311	16
Mean rank	27,992.34	535.47
Rank-1 accuracy	0%	22.88%
Rank-5 accuracy	0%	34.75%

5.2 FACES: FROM PHOTOS TO EMOJI

For face images, we use a set $\mathbf{s}$ of one million random images without identity information. The set $\mathbf{t}$ consists of assorted facial avatars (*emoji*) created by an online service (`bitmoji.com`). The emoji images were processed by a fully automatic process that localizes, based on a set of heuristics, the center of the irides and the tip of the nose. Based on these coordinates, the emoji were centered and scaled into 152×152 RGB images.

As the function f, we employ the representation layer of the DeepFace network Taigman et al. (2014). 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 D takes 152×152 RGB images (either natural or scaled-up emoji) and consists of 6 blocks, each containing a convolution with stride 2, batch normalization, and a leaky ReLU with a parameter of 0.2. Network g maps f's 256D representations to 64×64 RGB images through a network with 5 blocks, each consisting of an upscaling convolution, batch-normalization and ReLU. Adding 1×1 convolution to each block resulted in lower L_{CONST} training errors, and made g 9-layers deep. We set $\alpha = 100$, $\beta = 1$, $\gamma = 0.05$ as the tradeoff hyper-parameters within L_G via validation. As expected, higher values of α resulted in better f-constancy, however introduced artifacts such as general noise or distortions. The network was trained for 3 epochs, the point where no further reduction of validation error was observed on L_{CONST}.

In order to upscale the 64×64 output to print quality, we used the method of Dong et al. (2015), which was shown to work well on art. We did not retrain this network for our application, and apply the published one to the final output of our method after its training was finished. Results without this upscale are shown, for comparison, in Appendix C.

Comparison With Human Annotators For evaluation purposes only, a team of professional annotators manually created an emoji, using a web service, for 118 random images from the CelebA dataset (Yang et al., 2015). Fig. 4 shows side by side samples of the original image, the human generated emoji and the emoji generated by the learned generator function G. As can be seen, the automatically generated emoji tend to be more informative, albeit less restrictive than the ones created manually.

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, which were not included in $\mathbf{s}$. We then employed the VGG face CNN descriptor of Parkhi et al. (2015) in order to perform retrieval as follows. For each image x in our manually annotated set, we create a gallery $\mathbf{s}' \cup x'$, where x' is the other image of the person in x. We then perform retrieval using the VGG face descriptor using either the manually created emoji or $G(x)$ as probe.

The VGG network is used in order to avoid a bias that might be caused by using f both for training the DTN and for evaluation. The results are reported in Tab. 4. As can be seen, the emoji generated by G are much more discriminative than the emoji created manually and obtain a median rank of 16 in cross-domain identification out of 10^5 distractors.

Multiple Images Per Person We evaluate the visual quality that is obtained per person and not just per image, by testing DTN on the Facescrub dataset (Ng & Winkler, 2014). For each person p, we considered the set of their images X_p, and selected the emoji that was most similar to their

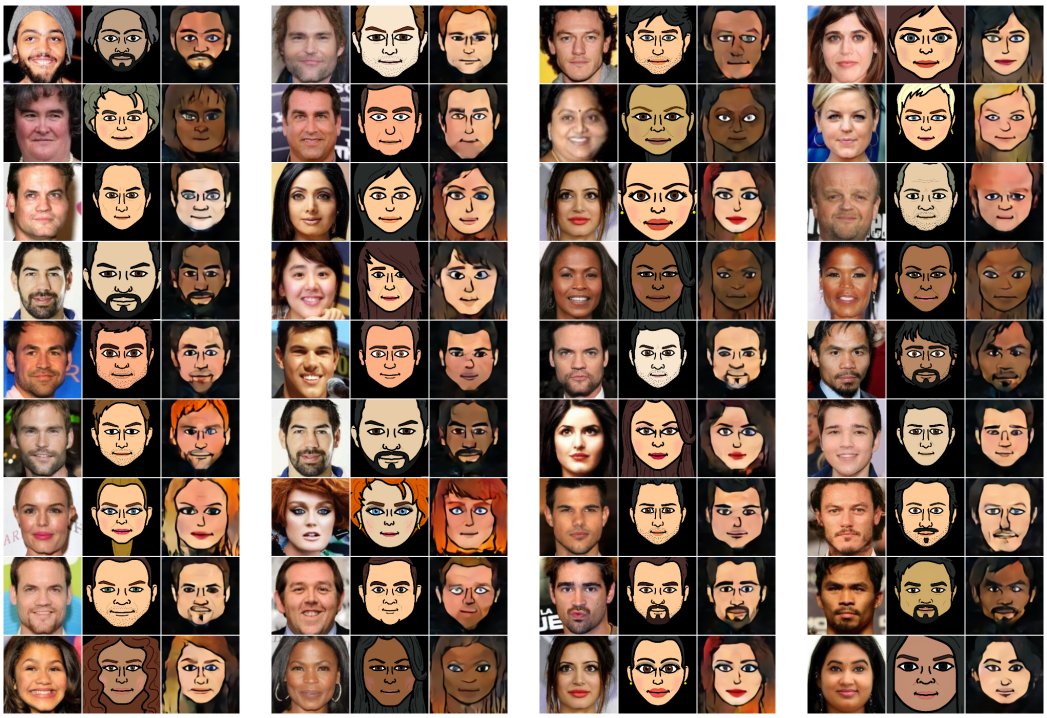


Figure 4: Shown, side by side are sample images from the CelebA dataset, the emoji images created manually using a web interface (for validation only), and the result of the unsupervised DTN. See Tab. 4 for retrieval performance.

source image:

$$\arg\min_{x \in X_p} ||f(x) - f(G(x))|| \tag{8}$$

This simple heuristic seems to work well in practice; The general problem of mapping a set $X \subset S$ to a single output in T is left for future work. Fig. 2(b) contains several examples from the Facescrub dataset. For the complete set of identities, see Appendix A.

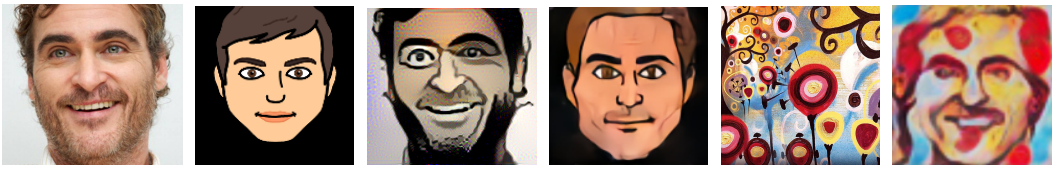
Transferring both identity and expression We also experimented with multiple expressions. As it turns out the face identification network f encodes enough expression information to support a successful transfer of both identity as well as expression, see Appendix B.

Network Visualization The obtained mapping g can serve as a visualization tool for studying the properties of the face representation. This is studied in Appendix D by computing the emoji generated for the standard basis of $\mathbb{R}^{256}$. The resulting images present a large amount of variability, indicating that g does not present a significant mode effect.

5.3 Style Transfer as a specific Domain Transfer task

Fig. 5(a-c) demonstrates that neural style transfer Gatys et al. (2016) cannot solve the photo to emoji transfer task in a convincing way. The output image is perhaps visually appealing; However, it does not belong to the space $\mathbf{t}$ of emoji. Our result are given in Fig. 5(d) for comparison. Note that DTN is able to fix the missing hair in the image.

Domain transfer is more general than style transfer in the sense that we can perform style transfer using a DTN. In order to show this, we have transformed, using the method of Johnson et al. (2016), the training images of CelebA based on the style of a single image (shown in Fig. 5(e)). The original photos were used as the set $\mathbf{s}$, and the transformed images were used as $\mathbf{t}$. Applying DTN, using face representation f, we obtained styled face images such as the one shown in the figure 5(f).



(a) (b) (c) (d) (e) (f)

Figure 5: Style transfer as a specific case of Domain Transfer. (a) The input content photo. (b) An emoji taken as the input style image. (c) The result of applying the style transfer method of Gatys et al. (2016). (d) The result of the emoji DTN. (e) Source image for style transfer. (f) The result, on the same input image, of a DTN trained to perform style transfer.

6 Discussion and Limitations

Asymmetry is central to our work. Not only does our solution handle the two domains S and T differently, the function f is unlikely to be equally effective in both domains since in most practical cases, f would be trained on samples from one domain. While an explicit domain adaptation step can be added in order to make f more effective on the second domain, we found it to be unnecessary. Adaptation of f occurs implicitly due to the application of D downstream.

Using the same function f, we can replace the roles of the two domains, S and T. For example, we can synthesize an SVHN image that resembles a given MNIST image, or synthesize a face that matches an emoji. As expected, this yields less appealing results due to the asymmetric nature of f and the lower information content in these new source domains, see Appendix E.

Domain transfer, as an unsupervised method, could prove useful across a wide variety of computational tasks. Here, we demonstrate the ability to use domain transfer in order to perform unsupervised domain adaptation. While this is currently only shown in a single experiment, the simplicity of performing domain adaptation and the fact that state of the art results were obtained effortlessly with a simple nearest neighbor classifier suggest it to be a promising direction for future research.

References

Shai Ben-david, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *NIPS*, pp. 137–144. 2006.

Andrew Brock, Theodore Lim, and Nick Ritchie, J. M.and Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*, pp. 767–774. 2012.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *arXiv preprint arXiv:1501.00092*, 2015.

Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *CoRR*, abs/1602.02644, 2016.

Gilles Fauconnier and Mark Turner. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books, 2003.

Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, pp. 2960–2967, 2013.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030, January 2016.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pp. 2672–2680. 2014.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *The International Conference on Learning Representations (ICLR)*, 2016.

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*, 19(5):866–885, 2013.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

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

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

H.W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *Proc. IEEE International Conference on Image Processing (ICIP), Paris, France*, 2014.

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

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

Leonid I. Rudin, Stanley Osher, and Emad 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*, pp. 259–268, 1992.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

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

Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *ICCV*, pp. 3676–3684, 2015.

Andrey Zhmoginov and Mark Sandler. Inverting face embeddings with convolutional neural networks. *arXiv preprint arXiv:1606.04189*, 2016.

A FACESCRUB DATASET GENERATIONS

In Fig. 6 we show the full set of identities of the Facescrub dataset, and their corresponding generated emoji.

Figure 6: All 80 identities of the Facescrub dataset. The even columns show the results obtained for the images in the odd column to the left. Best viewed in color and zoom.

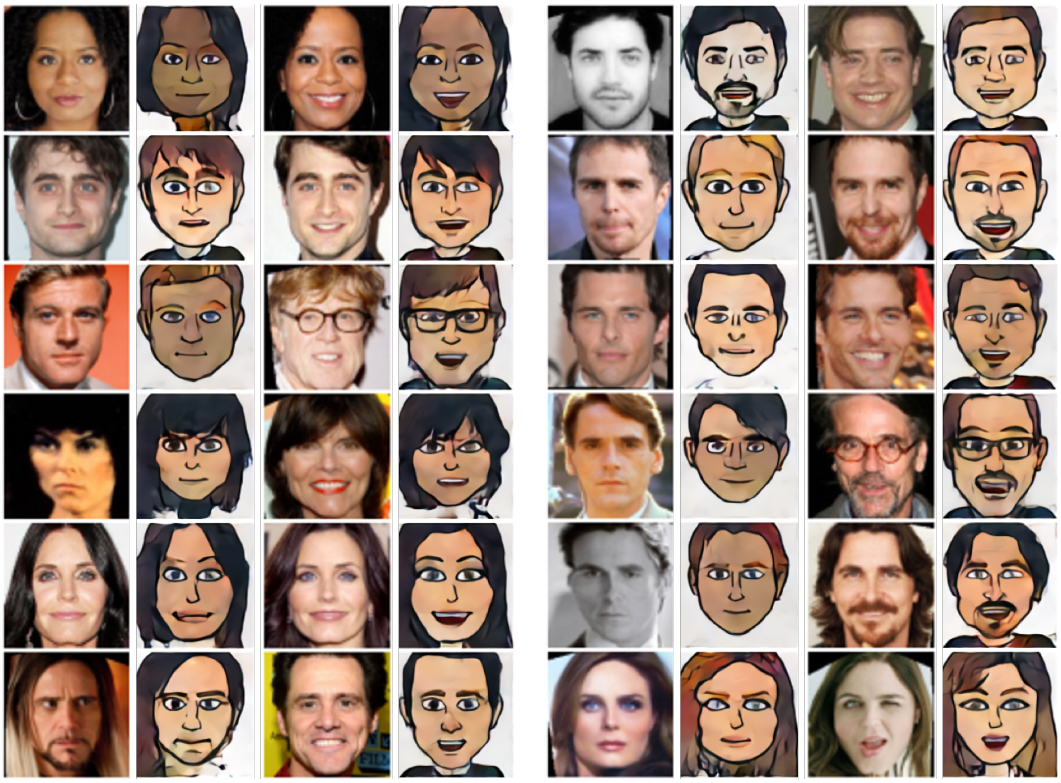


Figure 7: Maintaining expression in the domain transfer. In order to support a smiling expression, random smiling emoji were added to set of unlabeled samples from the domain T and the DTN was re-trained. Each quadruplet include two pairs of $\{$face, emoji$\}$ of the *same* identity in the two modes respectively: not-smiling and smiling. Odd columns are input; Subsequent even columns are output.

B TRANSFERRING NON-IDENTITY DATA

f may encode, in addition to identity, other data that is desirable to transfer. In the example of faces, this information might include expression, facial hair, glasses, pose, etc. In order to transfer such information, it is important that the set of samples in the target domain $\mathbf{t}$ present variability along the desirable dimensions. Otherwise, the GAN applied in the target domain (Eq. 4) would maintain these dimensions fixed. The set $\mathbf{t}$ employed throughout our experiments in Sec. 5.2 was constructed by sampling emoji of neutral expression. To support a smiling expression for example, we simply added to set t random smiling emoji and re-trained the DTN. The results, presented in Fig. 7, demonstrate that f contains expression information in addition to identity information, and that this information is enough in order to transfer smiling photos to smiling emoji.

C THE EFFECT OF SUPER-RESOLUTION

As mentioned in Sec. 5, in order to upscale the 64×64 output to print quality, the method of Dong et al. (2015) is used. Fig. 8 shows the effect of applying this postprocessing step.

D THE BASIS ELEMENTS OF THE FACE REPRESENTATION

Fig. 9 depicts the face emoji generated by g for the standard basis of the face representation (Taigman et al., 2014), viewed as the vector space $\mathbb{R}^{256}$.

E DOMAIN TRANSFER IN THE REVERSE DIRECTION

For completion, we present, in Fig. 10 results obtained by performing domain transfer using DTNs in the reverse direction of the one reported in Sec. 5.

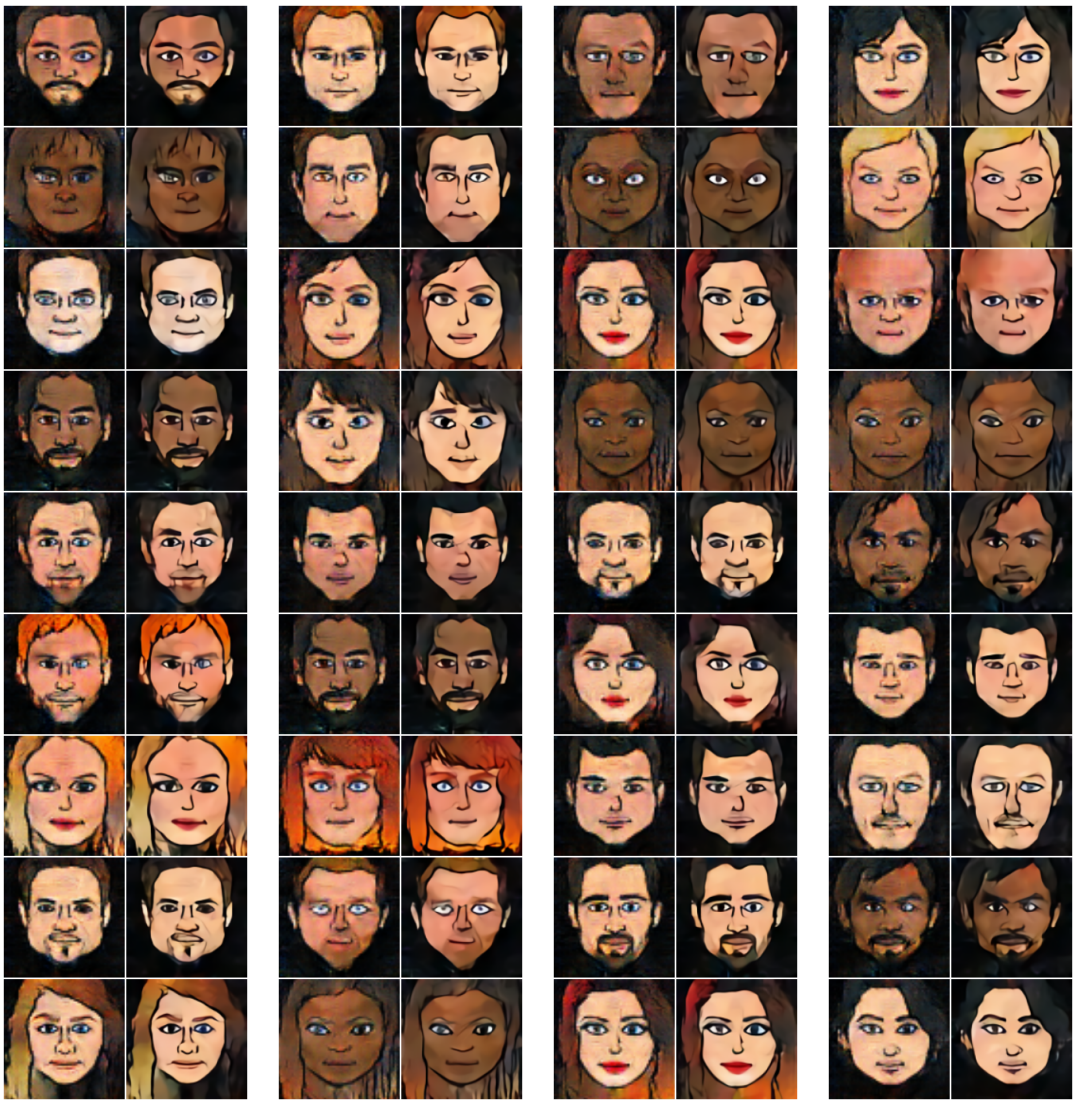


Figure 8: The images in Fig. 4 above with (right version) and without (left version) applying super-resolution. Best viewed on screen.

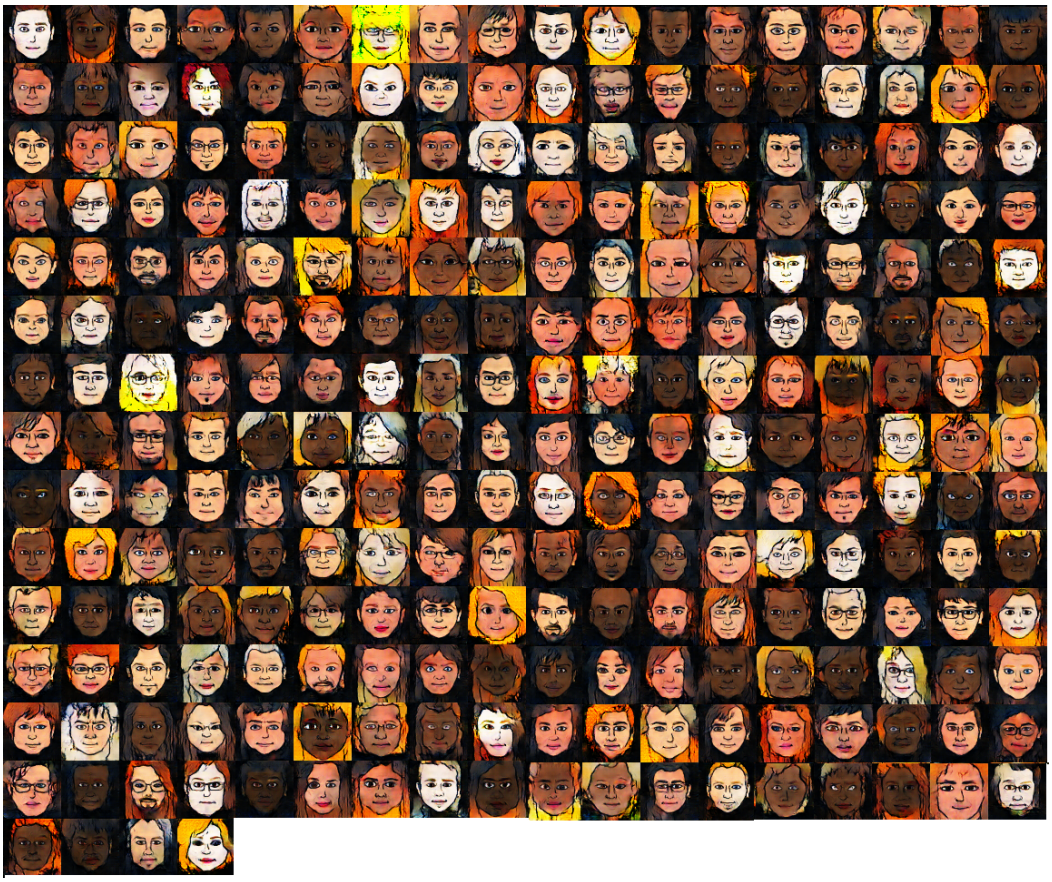
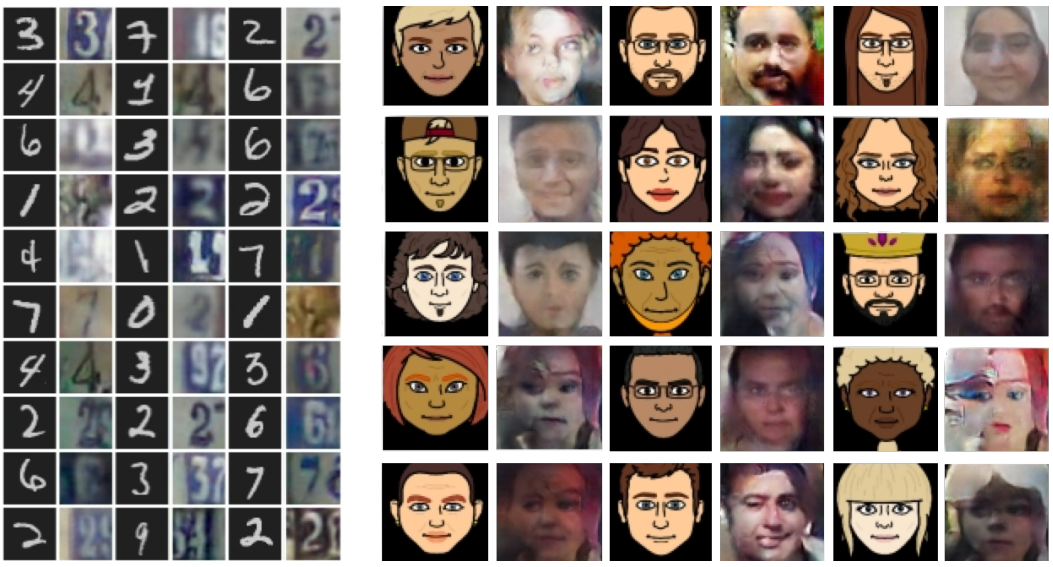


Figure 9: The emoji visualization of the standard basis vectors in the space of the face representation, i.e., $g(e_1)$,...,$g(e_{256})$, where e_i is the i standard basis vector in $\mathbb{R}^{256}$.



(a) (b)

Figure 10: Domain transfer in the other direction (see limitations in Sec. 6). Input (output) in odd (even) columns. (a) Transfer from MNIST to SVHN. (b) Transfer from emoji to face photos.