Towards Automatic Image Editing: Learning to See another You

Amir Ghodrati*[1]
http://homes.esat.kuleuven.be/~aghodrat/
Xu Jia*[1]
http://homes.esat.kuleuven.be/~xjia/
Marco Pedersoli[2]
marco.pedersoli@inria.fr
Tinne Tuytelaars[1]
http://homes.esat.kuleuven.be/~tuytelaa/

[1] ESAT-PSI
KU Leuven, iMinds
Leuven, Belgium

[2] THOTH
INRIA Grenoble
Grenoble, France

Abstract

In this paper we propose a method that aims at automatically editing an image by altering its attributes. More specifically, given an image of a certain class (*e.g.* a human face), the method should generate a new image as similar as possible to the given one, but with an altered visual attribute (*e.g.* the same face with a new pose or a different illumination). To this end, we propose a solution following an encoder-decoder pipeline. The desired attribute and the input image are independently encoded into a convolutional network and fused at feature map level. A convolutional decoder is then used to generate the target image. The result is further refined with another convolutional encoder-decoder network with the initial result and the original image as inputs. We evaluate the proposed method on MultiPIE dataset for three sub-tasks, that is, rotating faces, changing illumination and image inpainting. We show that the method is able to generate realistic images for the three tasks.

1 Introduction

When looking at pictures of family and friends, does any of the following comments sound familiar to you? *"Nice picture, just a pity her eyes are hidden behind those big sunglasses."* or *"I look so old on this picture - wish I could make myself look 5 years younger!"* or *"What would he look like if he grew a beard?"*. Now imagine the next generation of image editing tools, which are able not just to correct red eyes, but also modify specific attributes of the people depicted in photographs: remove their sunglasses, make them look younger or add that beard, while keeping all other facial characteristics and imaging conditions constant.

Thoughts along these lines motivated us to start looking into this problem. While our current method cannot yet be applied 'in the wild' in applications like the one sketched above, it does show promising results which make us think that automatic image editing might actually become doable in the near future. What we can do now is shown in Figure 1.

There have been several works on image generation [1, 2, 4, 6, 9, 10, 12, 14, 20, 25, 26, 27, 28, 29]. However, note how our problem setting is different from the one tackled by

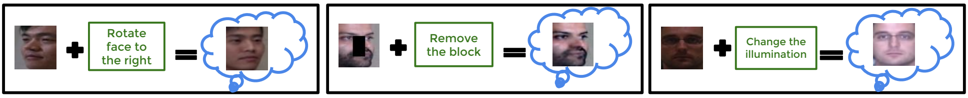


Figure 1: Image generation for three different tasks using the proposed method: (**Left**) rotating faces, (**Middle**) image inpainting and (**Right**) changing the illumination.

most image generation methods found in the literature. [2, 4, 6, 14] train a model to generate images from scratch, i.e. without an input image as reference. Others often focus only on changing a face or object pose (e.g. [1, 26]), or learn to generate faces only for a canonical setting (e.g. looking straight into the camera, with standard diffuse illumination and neutral expression [27, 28]).

We start with a large dataset of cropped and aligned faces, with a variety of attributes (e.g. pose and illumination) annotated and varied systematically for each individual in the database. We propose a model following the encoder-decoder fashion. It takes a face image as input and encodes it into several feature maps; takes a desired attribute vector as input and encodes it into several feature maps; then combines and deeply fuse these two flows of information; finally generates a new image with a convolutional decoder module. In addition, in order to generate more realistic images we adopt a coarse-to-fine scheme, dividing the problem in two stages with each one focusing on one aspect. The first stage is in charge of rendering a global representation of the desired object, while the second focuses on local refinements to remove some artifacts. To demonstrate the effectiveness of the proposed method, we evaluate it on the MultiPIE [7] dataset for three sub-tasks, that is, changing the face pose, changing the image illumination and also image inpainting.

The qualitative and quantitative results show that the proposed method can generate face images of good quality and keep all information of the input face except what is specified by the desired attribute. We also evaluate the proposed method on face retrieval task. Given a query image we want to retrieve similar images but with some altered attributes, similar to what is done in [5]. Instead of learning an attribute detector first, we generate with our method an image with the desired attribute and then take the altered image as query to do standard similarity-based retrieval.

In summary, the main contributions of this paper are: i) definition of a new problem, where the goal is to generate images as similar as possible to a source image yet with one attribute changed; ii) a solution that follows an encoder-decoder pipeline, where the desired attribute modification is first encoded then integrated at feature map level; iii) the insight that the result can be refined by adding another convolutional encoder-decoder model; and iv) good qualitative and quantitative results on different tasks on MultiPIE dataset.

The remainder of this paper is organised as follows. First, in section 2, we discuss related work. Section 3 describes our proposed method. Section 4 details the experimental evaluation of our method, and section 5 concludes the paper.

2 Related Work

Recently, there have been several works addressing the task of image generation [1, 2, 4, 6, 9, 10, 12, 14, 19, 20, 24, 25, 26, 27, 28, 29]. These methods can roughly be divided into two categories.

A first line of works follows an unsupervised approach. Some of these methods learn the distribution of the training images and generate an image from scratch [2, 4, 6, 14, 19]. Hin-

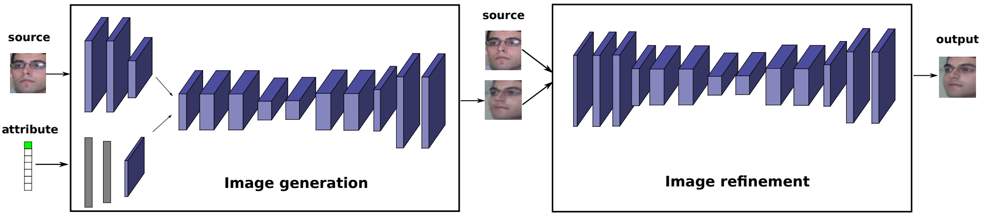


Figure 2: An overview of our proposed method. Given a source image and attribute vector, we modify the source based on the attribute vector and generate an image in a two stage approach.

ton *et al.* [9] and Tieleman [25] proposed the capsule network to disentangle different visual components in the code layer of an autoencoder. Kingma and Welling [10] and Rezende *et al.* [20] proposed the variation autoencoder. It applies the re-parameterization trick to the latent variables in the code layer to model different factors of the visual appearance. Kulkarni *et al.* [12] proposed to use convolutional variational autoencoder and a special training scheme to learn interpretable graphics code for different appearance transformations.

The second group of works generates images conditioned on either an image or several attributes. Dosovitskiy *et al.* [1] map high-level information such as chair type and viewpoint into the 2D image space to generate different chair images. Radford *et al.* [19] propose guidelines for stable training of deep convolutional Generative Adversarial Network (GANs). They generate faces conditioned on desired attributes, however, it is not guaranteed that their method always generates semantically meaningful images as they did not explicitly train their model for this task. Gardner *et al.* [3] find a traversal path on the manifold of natural images from input image towards the target image with desired class labels. Zhu *et al.* [28, 29], Yim *et al.* [27] and Yang *et al.* [26] propose to generate images conditioned on an image and an attribute. Given an object in a specific pose, [28] propose a network to generate frontal faces with fixed illumination. Yim *et al.* [27] propose to generate a face with desired pose and neutral illumination with a multi-task model which includes a generation DNN and a reconstruction DNN. Yang *et al.* [26] use a recurrent network to model pose changes of faces. Our method also falls in this category and focuses on the generation of face image. However in contrast to the previously mentioned approaches, our method is not designed specifically for changing the pose of a face; we will show the generality of our network on other tasks. In addition, most methods will change other aspects of a face (*e.g.* illumination or identity) during generation; while in our task we want to generate an image with the desired attribute while preserving the other image information as before.

3 Proposed Method

In this section, we describe the proposed convolutional encoder-decoder architecture for image generation. The overview of the proposed method is shown in Figure 2. Given a source image and a target attribute vector, our goal is to generate a new image with the target attribute while maintaining as much as possible other aspects of the appearance of the source image. To this end, we propose a two-stage approach, with a first network for image generation (see 3.1) and a second one for image refinement (see 3.2). Let us denote the source image as $\mathbf{X}$ and the target image and the target attribute as $\mathbf{Y}$ and $\mathbf{C_Y}$.

3.1 Convolutional Encoder-Decoder Architecture for Image Generation

Inspired by the recent success of deconvolutional networks in generating accurate images of chairs from high-level descriptions [1] and semantic segmentation [16, 18], we adopt a convolutional encoder-decoder architecture for our task. However, in contrast to previous works we deal with two inputs coming from different modalities: an image and a target attribute vector. The architecture can be conceptually divided into four operations: image encoding, attribute vector encoding, feature map fusion and image decoding.

Image encoding. In this component, we encode a source image $\mathbf{X}$ into a set of feature maps via two convolutional layers and a max-pooling layer. Inspired by the OxfordNet [22] we use two consecutive convolutional layers, each of which includes filters of size (3×3) and one rectifier linear unit (ReLU) [11]. The two consecutive convolutional layers help to increase the receptive field with a reduced increment of the number of parameters. The convolutional layers are able to capture local information of the source image and later we will show how to use them to generate the target image. The structure of this part can be expressed as Conv$(3,3,64)$-ReLU-Conv$(3,3,64)$-ReLU-Pool$(2,2)$. Since we use a $60 \times 60 \times 3$ as input, we get feature maps of size $30 \times 30 \times 64$ at the end.

Attribute vector encoding. We express the attribute as a one-hot vector. In order to convert the knowledge contained in the attribute vector into visual information we convert it into feature maps which can be easily combined with the convolutional features of the input image. We apply two fully connected layers to the one-hot vector and then reshape it to feature maps of size $30 \times 30 \times n_a$, where n_a is the number of values for an attribute. The structure of this part can be expressed as FC(512)-FC($900 \times n_a$)-Reshape$(30,30,n_a)$.

Feature map fusion. In this step, we fuse the feature maps obtained from the encoding of the input image and the attribute vector. The feature maps with attribute information can propagate their message to the feature maps extracted from the source input. The two sets of feature maps are first stacked together, generating a new set of feature maps of $30 \times 30 \times (64+n_a)$. Then a feature map fusion layer similar to the cross channel pooling layer in [15, 23] is applied on top to fuse the two sets of feature maps. The cross channel pooling layer can be viewed as a $(1,1)$ convolutional layer followed by a ReLU. In our case, the feature map fusion layer is more sophisticated, including several consecutive convolutional layers. The convolution operations compute the weighted average of feature maps so that it allows sufficient interaction between feature maps of the source image and the attributes in a learning framework. The structure of this part can be expressed as Concat-Conv$(3,3,128)$-ReLU-Conv$(3,3,128)$-ReLU-Pool$(2,2)$-Conv$(1,1,128)$-ReLU. The output of this step is a feature map of size $15 \times 15 \times 128$.

Image decoding. Once the feature maps computed from source image $\mathbf{X}$ and the attribute vector $\mathbf{C_Y}$ have been fully integrated, the next step is to generate an image with the same size as the input image. This module aggregates local information from different feature maps. Similar to [1], a deconvolution module here consists of a 2×2 unpooling layer and two convolutional layers. The unpooling layer doubles the size of feature maps in the previous layer by replacing each element of the feature map with a 2×2 block. The top left corner is filled in with the element of the feature map and the rest are filled with zeros. The unpooling layer is followed by two consecutive $(3,3)$ convolutional layers. Note that all convolutional layers have a ReLU activation except the last one because the output should have both positive and negative values after input images being normalized to zero mean and one standard deviation. The structure of this part can be expressed as Unpool$(2,2)$-Conv$(3,3,64)$-ReLU-Conv$(3,3,1)$-ReLU-Unpool$(2,2)$-Conv$(3,3,64)$-Conv$(3,3,1)$. The output of this step is an image of size $60 \times 60 \times 3$.

3.2 Image Generation Refinement

The ideal generated image should have good quality, have the desired target attribute and keep the appearance of the input image. It is difficult for a single network to generate an image that satisfies all the above requirements simultaneously. Therefore, we adopt the divide-and-conquer scheme, dividing the problem into two stages. The output of the proposed convolutional encoder-decoder network produces already a reasonable result, but it still has some missing details and some artifacts. As shown in Figure 2, we propose to add another convolutional encoder-decoder network to perform image refinement in the second stage. The second stage takes as input the source image and the generated image of the first stage. These two inputs are first concatenated channel-wise. Then we apply several convolutional, ReLU and max-pooling layers in the encoding process followed by unpooling, convolutional and ReLU layers in the decoding process. Convolutional layers locally fuse the information from two inputs and refine the output of the first stage network. Also, note that for decoding the image, we use separate set of parameters than image encoding. The architecture of the whole network is as follows: Concat-Conv$(3,3,64)$-ReLU-Conv$(3,3,64)$-ReLU-Pool$(2,2)$-Conv$(3,3,128)$-ReLU-Conv$(3,3,128)$-Pool$(2,2)$-Conv$(1,1,128)$-ReLU-Unpool$(2,2)$- Conv $(3,3,128)$-ReLU-Conv$(3,3,64)$-ReLU-Unpool$(2,2)$-Conv$(3,3,64)$-Conv$(3,3,1)$. Our method shares similar philosophy as the coarse-to-fine two stage method used for synthesizing novel faces [17]. In [17], they first use a global parametric model to make an approximate estimation of the global structure of a face. Then a local non-parametric model is conditioned on the global estimation to refine the initial result of the global model. In our framework, the goal of the first-stage network is to generate an image that has the correct global structure of the target face. In the second stage, the goal is to keep the visual consistency with the outputs of the first stage but removing artifacts and adding the details to the face. We show the effectiveness of such paradigm for image generation in section 4.

3.3 Training

We train the two networks successively. The input of the first network are the source image of size $60 \times 60 \times 3$ and a one-hot attribute vector. Also, in each network the encoder and decoder have their own, separate parameters and are trained together. own, separate parameters and Each image is preprocessed by subtracting the mean and dividing by the standard deviation. Then, the output of the first network and the source image are fed as input to the second network. For both networks, we use Mean Squared Error (MSE) as the loss function. The training is carried out using mini-batch gradient descent with backpropagation [13] and the batch size is set to 32. We use a fixed momentum of 0.95 and learning rate of $1e-6$. All the weights are initialized with the method proposed in [8].

4 Experiments

In order to demonstrate the effectiveness of the proposed method we evaluate it for three different tasks. The main task is to rotate the face and is carried out on MultiPIE dataset [7].

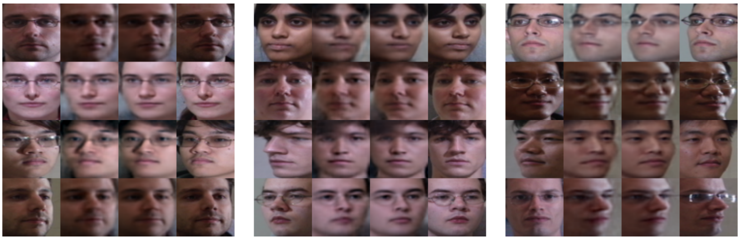


Figure 3: Some qualitative results of our image generation from test data of MultiPIE. In each row, first column is input image, last column is ground-truth target image, 2nd column is output of first stage and 3rd column is generated image of second stage network.

We extensively evaluate our method for this task, showing both qualitative and quantitative results. The other two tasks are generating faces with different illumination on MultiPIE dataset and filling in the missing part for a face image on synthetic data generated from MultiPIE. We show some qualitative results for these two tasks.

4.1 Rotating Faces

The session 1 of MultiPIE dataset consists of images of 249 identities under 15 poses and 20 different illumination conditions. We select a subset of it that covers 7 poses ($-45°$ to $+45°$). The first 100 subjects are used for training and the rest are used for testing. All faces are aligned and cropped based on eyes and chin annotations provided by Shafey *et al.* [21], then resized to 60×60 pixels.

Here we consider pose as an attribute and aim at generating faces with the same identity and illumination as the input image but with the desired pose. The input to our method is an image and a target pose vector. To this end, during training we build a set of image pairs, where the images show the same person, with the same illumination, but with different pose (i.e. $100 \times 20 \times P_7^2 = 84000$ pairs). The first element of a pair is considered as input image of the network and the second element is the ground-truth target image.

Image Generation Results. In Figure 3 we show some qualitative results, i.e., the generated images from both stages of the method. We can see the proposed method can generate face images that are visually similar to the target face. Notice how the generated faces from the second stage have better image quality and details than first stage output.

We qualitatively compare our method with the method of [27] in Figure 4. From the figure, we can see that our method (left) preserves the source image information better than theirs (right), that is, more details of the faces but less clutter and noise. However, note that the two approaches have been trained for different tasks; the network in [27] is trained to generate neutral illumination regardless of the input image illumination, while ours aims at keeping the same illumination as in the original image.

It is worth mentioning that [27] is much more complex than ours. Due to the use of locally linear and fully connected layers, their network has much more parameters than ours (200M vs. 6M parameters). Besides, they add an auxiliary branch for reconstruction in their network to preserve the identity. In contrast, we have a much simpler architecture, which only contains fully connected layers for attribute vector encoding and only convolutional layers elsewhere. The attribute vector encoding module makes a rough estimation about the desired pose. This information is propagated to the later part of the network and deeply

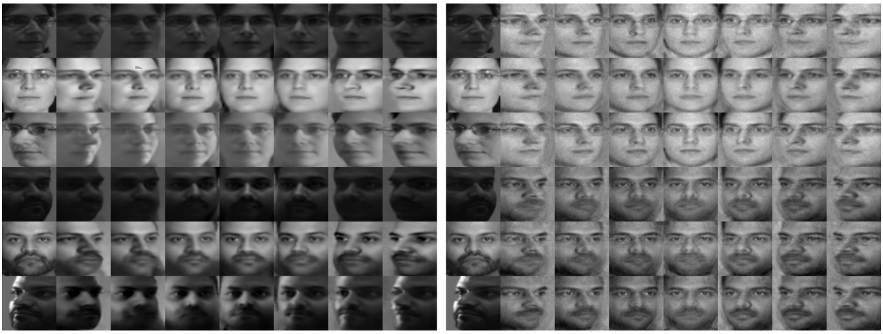


Figure 4: Qualitative comparison between our generated images (**Left**) and [27] (**Right**). On each set the first column is the input face and next 7 ones are generated faces in different poses. We convert our generated faces to gray-scale for fair comparison with [27].

per-pixel MSE	First stage	Second stage	CPI [27]
All	381.5	376.3	–
Neutral illumination subset	578.5	570.5	884.4

Table 1: Mean Squared Error on all images and on a subset of 510 images with neutral illumination, for a fair comparison with [27].

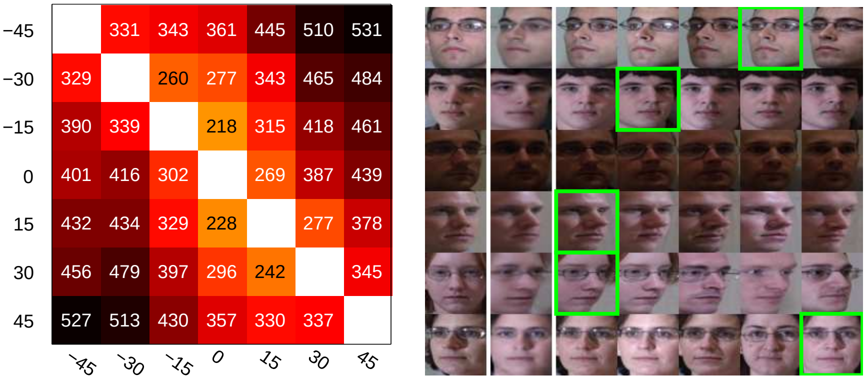
fused with the feature maps computed from the input image which contains both identity and illumination information.

We quantitatively validate the effectiveness of our method as well. We randomly select 10000 generated images and compare the performance of each stage in terms of per-pixel mean squared error (MSE) between generation and ground-truth image and report it in Table 1 (first row). The table confirms the visual impression that the generation of the second stage generates higher quality images. This implies that the second stage network works as expected: it locally refines the initial generation via additional convolutional layers. We also compare our network with [27] in terms of per-pixel MSE. To this end the authors kindly provided us the CPI features (images) that they have used for face recognition in their paper. To have a fair comparison, we only select a subset of test faces with neutral illumination (510 faces) since their network is designed to be illumination-invariant (i.e. it always generates faces with neutral illumination). From Table 1 (second row), we observe that our method has a better per-pixel MSE than [27], which again confirms the visual impression.

To get some insight about the performance under various pose changes, we compute per-pixel MSE for different pose rotations in Figure 5 (**Left**). As is expected, small pose changes are easier to deal with than large ones. Besides, we also observe that the task is easier when the input and output are symmetric poses (e.g., compare the result from -15 to 15 with the one from -15 to -45) and when the desired pose is closer to frontal face (e.g., compare the MSE from -30 to 0 with the one from 0 to +30).

In addition, in Figure 6 we divide the generated test data into 7 groups based on the desired face pose. For each group we compute per-pixel MSE separately. As is shown in the figure, most large pixel errors come from the silhouette of the face and hair and neck regions since there is much uncertainty and it is difficult to infer that from single image.

Retrieval results. To demonstrate how our method preserves other aspects of a face (e.g., identity and illumination) while changing the pose, we conduct an experiment on face retrieval. We generate an image conditioned on the query image and the target pose using the proposed method and then find the K nearest neighbors for the generated face. In our



	-45	-30	-15	0	15	30	45
-45		331	343	361	445	510	531
-30	329		260	277	343	465	484
-15	390	339		218	315	418	461
0	401	416	302		269	387	439
15	432	434	329	228		277	378
30	456	479	397	296	242		345
45	527	513	430	357	330	337	

Figure 5: (**Left**) per-pixel MSE for various pose changes. The vertical axis is source pose degree and the horizontal is target pose degree.(**Right**) Some visual retrieval results using our method. The first column is the query image, the second is the generated image using our method and the remaining columns are the top 5 retrieval results. The green boxes are correct retrieved faces. A retrieved image is considered to be correct if its identity, pose and illumination are all correct.

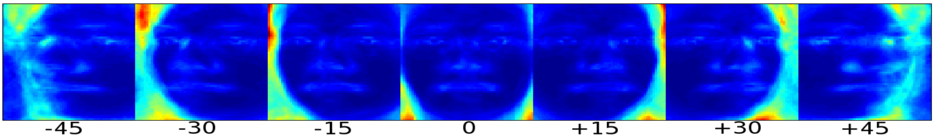


Figure 6: Visualization of per-pixel MSE. We grouped the generated faces based on their poses and for each group, we computed per-pixel MSE.

experiment we use L2-normalized features extracted from the 1×1 convolution layer of our second network and measure the distance of query and candidates using Euclidean distance. For the images to retrieve, as we do not know their pose, we pre-compute the features for all possible poses (7) and keep the one whose reconstruction is closer to the original image.

As evaluation criterion, we consider a retrieved image to be correct if its identity and its illumination are the same as the input and its pose is the desired one. The retrieval result for our method is 49.9% and 76.1% for recall@1 and recall@5 respectively. recall@K is defined as the percentage of queries for which the correct image is among the top K retrieved images. Some qualitative results of retrieval by generation are also shown in Figure 5 (Right).

4.2 Other Tasks

Changing illumination. We train another model to generate a face with desired illumination. To this end, we use the same dataset but changing the attribute vector to generate a face with specific illumination out of 20 different illumination conditions. In Figure 7, given an input (first column), we generate faces with different illumination by changing the attribute vector to the corresponding desired illumination. Quantitatively, the per-pixel MSE of test set is 193.3 and 146.6 respectively for the first and second stage. These numbers indicate this task is easier than rotating faces (see Table 1). This is expected since it is easier for convolutional filters to learn local illumination changes than propagating appearance information to distant locations. Besides, there is less uncertainty on the silhouette of a face.

Image Inpainting. We carry out another interesting experiment on the MultiPIE dataset

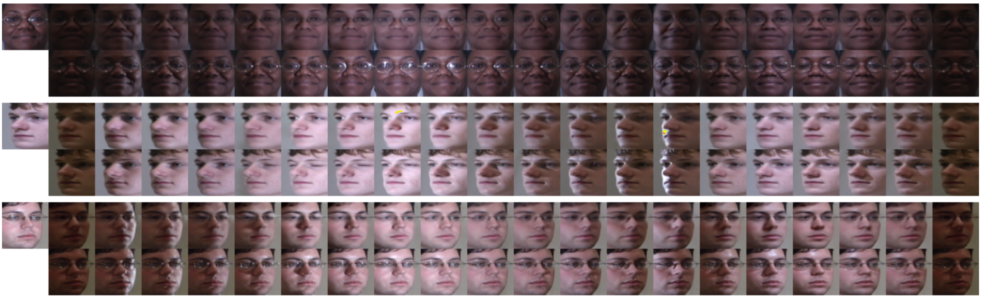


Figure 7: Qualitative results for the task of changing illumination. The most right image is input face. For each identity, the first row shows 20 faces generated under different illuminations and second row is the corresponding ground-truth face.

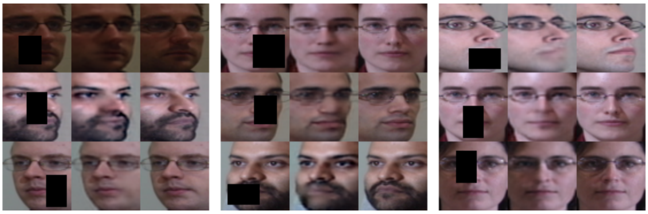


Figure 8: Qualitative results for the task of image inpainting. The first column shows the input image, the second column shows images generated with our method and the third column shows the complete image without the occluding pattern.

to demonstrate the ability of the proposed method in image generation. For this task, we randomly generate 10 black blocks of different shapes as occlusion patterns. For each image, one of these 10 patterns is selected and overlaid on the face image at a random location. We train the proposed model to learn to inpaint the occluded face image. In this case, the attribute vector is a binary value which specifies whether a face is occluded or not. As shown in Figure 8 our method can generate reasonably good images also for this task considering the high variability of the input image. The model fills in the occluded region using the knowledge it learned during training such as the continuity of local region and the symmetry within the face. The filled region is visually consistent with the non-occluded parts of the face. Quantitatively, the per-pixel MSE of test set is 64.6 for the second stage. These number indicates this task is also easier than rotating faces. However, we should notice that in this task, the modification is not global as two other tasks and just part of the image is desired to be altered. As an evidence, the MSE of source and target images is just 1473.0. This explain the small value of MSE error for this task.

5 Conclusion

In this work we define a new problem, where given an input image, the goal is to generate images with modified attributes while maintaining as much as possible the similarity to the original image. We propose a solution to this problem, for the case of cropped and aligned faces, in the form of a two stage encoder/decoder convolutional network, with the first stage for image generation and the second one for image refinement. We have validated our approach both qualitatively and quantitatively on MultiPIE dataset for three sub-tasks. For future work we would like to extend our method to address even more challenging scenarios like dealing with misaligned input faces and applying it to different object categories.

6 Acknowledgement

This work was supported by FWO through the project G.0696.12N " Representations and algorithms for captation, visualization and manipulation of moving 3D objects, subjects and scenes".

References

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

[2] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[3] Jacob R Gardner, Matt J Kusner, Yixuan Li, Paul Upchurch, Kilian Q Weinberger, and John E Hopcroft. Deep manifold traversal: Changing labels with convolutional features. *arXiv preprint arXiv:1511.06421*, 2015.

[4] J Gauthier. Conditional generative adversarial nets for convolutional face generation. Class Project forStanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester 2014, 2014.

[5] Amir Ghodrati, Xu Jia, Marco Pedersoli, and Tinne Tuytelaars. Swap retrieval: Retrieving images of cats when the query shows a dog. In *ICMR*, 2015.

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[7] Ralph Gross, Iain Matthews, Jeffrey F. Cohn, Takeo Kanade, and Simon Baker. Multipie. *Image Vision Comput.*, 28(5):807–813, 2010.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[9] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In *ICANN*, 2011.

[10] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[12] Tejas D. Kulkarni, Will Whitney, Pushmeet Kohli, and Joshua B. Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

[13] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[14] Yujia Li, Kevin Swersky, and Richard S. Zemel. Generative moment matching networks. In *ICML*, 2015.

[15] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[17] Umar Mohammed, Simon J. D. Prince, and Jan Kautz. Visio-lization: generating novel facial images. *ACM Trans. Graph.*, 2009.

[18] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

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

[20] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[21] Laurent El Shafey, Chris McCool, Roy Wallace, and Sébastien Marcel. A scalable formulation of probabilistic linear discriminant analysis: Applied to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1788–1794, 2013.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2014.

[24] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Single-view to multi-view: Reconstructing unseen views with a convolutional network. *arXiv preprint arXiv:1511.06702*, 2015.

[25] Tijmen Tieleman. *Optimizing neural networks that generate images*. PhD thesis, University of Toronto, 2014.

[26] Jimei Yang, Scott Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *NIPS*, 2015.

[27] Junho Yim, Heechul Jung, ByungIn Yoo, Changkyu Choi, Du-Sik Park, and Junmo Kim. Rotating your face using multi-task deep neural network. In *CVPR*, 2015.

[28] Zhenyao Zhu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning identity-preserving face space. In *ECCV*, 2013.

[29] Zhenyao Zhu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Multi-view perceptron: a deep model for learning face identity and view representations. In *NIPS*, 2014.