

Visual Data Augmentation through Learning

Grigorios G. Chrysos¹, Yannis Panagakis^{1,2}, Stefanos Zafeiriou¹

¹ Department of Computing, Imperial College London, UK

² Department of Computer Science, Middlesex University London, UK

{g.chrysos, i.panagakis, s.zafeiriou}@imperial.ac.uk

Abstract

The rapid progress in machine learning methods has been empowered by i) huge datasets that have been collected and annotated, ii) improved engineering (e.g. data pre-processing/normalization). The existing datasets typically include several million samples, which constitutes their extension a colossal task. In addition, the state-of-the-art data-driven methods demand a vast amount of data, hence a standard engineering trick employed is artificial data augmentation for instance by adding into the data cropped and (affinely) transformed images. However, this approach does not correspond to any change in the natural 3D scene. We propose instead to perform data augmentation through learning realistic local transformations. We learn a forward and an inverse transformation that maps an image from the high-dimensional space of pixel intensities to a latent space which varies (approximately) linearly with the latent space of a realistically transformed version of the image. Such transformed images can be considered two successive frames in a video. Next, we utilize these transformations to learn a linear model that modifies the latent spaces and then use the inverse transformation to synthesize a new image. We argue that this procedure produces powerful invariant representations. We perform both qualitative and quantitative experiments that demonstrate our proposed method creates new realistic images.

1. Introduction

The lack of training data has till recently been an impediment for training machine learning methods. The latest breakthroughs of Neural Networks (NNs) can be partly attributed to the increased amount of (public) databases with massive number of labels/meta-data. Nevertheless, the state-of-the-art networks include tens or hundreds of millions of parameters [8, 37], i.e. they require more labelled examples than we have available. To ameliorate the lack of sufficient labelled examples, different data augmentation methods have become commonplace during training. In this

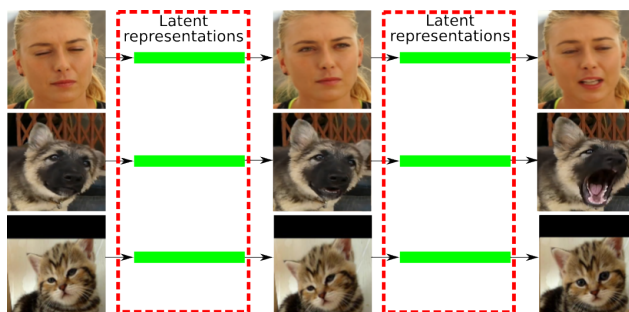


Figure 1: (Preferably viewed in color) We want to augment arbitrary images (left column) by learning local transformations. We find a low-dimensional space and learn the forward and inverse transformations from the image to the representation space. Then, we can perform a simple linear transformation in the (approximately) linear low-dimensional space and acquire a new synthesized image (the middle column). The same procedure can be repeated with the latest synthesized image (e.g. from the middle to the right columns).

work, we propose a new data augmentation technique that finds a low-dimensional space in which performing a simple linear change results in a nonlinear change in the image space.

Data augmentation methods are used as label-preserving transformations with twofold goals: a) avoid over-fitting, b) ensure that enough samples have been provided to the network for learning. A plethora of label-preserving transformations have been proposed, however the majority is classified into a) either model-based methods, b) or generic augmentations. The model-based demand an elaborate model to augment the data, e.g. the 3DMM-based [2] face profiling of [29], the novel-view synthesis from 3D models [26]. Such models are available for only a small number of classes and the realistic generation from 3D models/synthetic data is still an open problem [3]. The second augmentation category is comprised of methods defined artificially; these methods do not correspond to any natural

movement in the scene/object. For instance, a 2D image rotation does not correspond to any actual change in the 3D scene space; it is purely a computational method for encouraging rotation invariance.

We argue that a third category of augmentations consists of local transformations. We learn a nonlinear transformation that maps the image to a low-dimensional space that is assumed to be (approximately) linear. This linear property allows us to perform a linear transformation and map the original latent representation to the representation of a slightly transformed image (e.g. a pair of successive frames in a video). If we can learn the inverse transform, i.e. mapping from the low-dimensional space to the transformed image, then we can modify the latent representation of the image linearly and this results in a nonlinear change in the image domain.

We propose a three-stage approach that learns a forward transformation (from image to low-dimensional representation) and an inverse transformation (from latent to image representation) so that a linear change in the latent space results in a nonlinear change in the image space. The forward and the inverse learned transformations are approximated by an Adversarial Autoencoder and a GAN respectively.

In our work, we learn object-specific transformations while we do not introduce any temporal smoothness. Even though learning a generic model for all classes is theoretically plausible, we advocate that with the existing methods, there is not sufficient capacity to learn such generic transformations for all the objects. Instead we introduce object-specific transformations. Even though we have not explicitly constrained our low-dimensional space to be temporally smooth, e.g. by using the cosine distance, we have observed that the transformations learned are powerful enough to linearize the space. As a visual illustration, we have run T-SNE [17] with the latent representations of the first video of 300VW [28] against the rest 49 videos of the published training set; Fig. 2 validates our hypothesis that the latent representations of that video reside in a discrete cluster over the rest of the representations. In a similar experiment with the collected videos of cats, the same conclusion is reached, i.e. the representations of the first video form a discrete cluster.

We have opted to report the results in the facial space that is highly nonlinear, while the representations are quite rich. To assess further our approach, we have used two ad-hoc objects, i.e. cat faces, dog faces, that have far less data labelled available online. Additionally, in both ad-hoc objects the shape/appearance presents greater variation than that of human faces, hence more elaborate transformations should be learned.

In the following Sections we review the neural networks based on which we have developed our method (Sec. 2.1), introduce our method in Sec. 3. Sequentially, we demon-

strate our experimental results in Sec. 4. Due to the restricted space, additional visualizations are deferred to the supplementary material, including indicative figures of the cats’, dogs’ videos, additional (animated) visual results of our method, an experiment illustrating that few images suffice to learn object-specific deformable models. We strongly encourage the reviewers to check the supplementary material.

Notation: A small (capital) bold letter represents a vector (matrix); a plain letter designates a scalar number. A vectorized image of a dynamic scene at time t is denoted as $\mathbf{i}^{(t)}$, while $\mathbf{i}_k^{(t_k)}$ refers to the k^{th} training sample.

2. Background

The following lines of research are related with our proposed method:

Model-based augmentation for faces: The methods in this category utilize 2D/3D geometric information. In T-CNN [32] the authors introduce an alignment-sensitive method tailored to their task. Namely, they warp a face from its original shape (2D landmarks) to a similar shape (based on their devised clustering). Recently, Zhu *et al.* [36] use a 3D morphable model (3DMM) [2] to simulate the effect of profiling for synthesizing images in large poses. Tran *et al.* in [29] fit a 3DMM to estimate the facial pose and learn a GAN conditioned on the pose. During inference, new facial images are synthesized by sampling different poses. The major limitation of the model-based methods is that they require elaborate 2D/3D models. Such models have been studied only for the human face¹ or the human body, while the rest objects, e.g. animals faces, have not attracted such attention yet. On the contrary, our method is not limited to any object (we have learned models with cats’ faces and

¹18 years since the original 3DMM model and the problem is not solved for all cases.

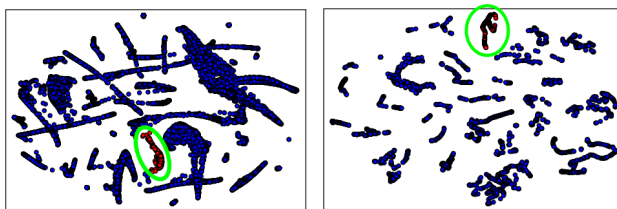


Figure 2: (Preferably viewed in color) T-SNE [17] in the latent representations of a) 300VW [28] (left Fig.), b) cats’ videos. In both cases the representations of the first video (red dots) are compared against the rest videos (blue dots)). To avoid cluttering the graphs every second frame is skipped (their representation is similar to the previous/next frame). For further emphasis, a green circle is drawn around the red points.

dogs’ faces) and does not require elaborate 3D/2D shape models.

Unconditional image synthesis: The successful application of GANs [5] in a variety of tasks including photo-realistic image synthesis [14], style transfer [34], inpainting [25], image-to-image mapping tasks [11] has led to a proliferation of works on unconditional image synthesis [1, 35]. Even though unconditional image generation has significant applications, it cannot be used for conditional generation when labels are available. Another line of research is directly approximating the conditional distribution over pixels [23]. The generation of a single pixel is conditioned on all the previously generated pixels. Even though realistic samples are produced, it is costly to sample from them; additionally such models do not provide access to the latent representation.

Video frames’ prediction: The recent (experimental) breakthroughs of generative models have accelerated the progress in video frames prediction. In [30] the authors learn a model that captures the scene dynamics and synthesizes new frames. To generalize the deterministic prediction of [30], the authors of [33] propose a probabilistic model, however they show only a single frame prediction in low-resolution objects. In addition, the unified latent code z (learned for all objects) does not allow particular motion patterns, e.g. of an object of interest in the video, to be distinguished. Lotter *et al.* [15] approach the task as a conditional generation. They employ a Recurrent Neural Network (RNN) to condition future frames on previously seen ones, which implicitly imposes temporal smoothness. A core differentiating factor of these approaches from our work is that they i) impose temporal smoothness, ii) make simplifying assumptions (e.g. stationary camera [30]); these restrictions constrain their solution space and allow for realistic video frames’ prediction. In addition, the techniques for future prediction often result in blurry frames, which can be attributed to the multimodal distributions of unconstrained natural images, however our end-goal consists in creating realistic images for highly-complex images, e.g. animals’ faces.

The work of [6] is the most similar to our work. The authors construct a customized architecture and loss to linearize the feature space and then perform frame prediction to demonstrate that they have successfully achieved the linearization. Their highly customized architecture (in comparison to our off-the-shelves networks) have not been applied to any highly nonlinear space, in [6] mostly synthetic, simple examples are demonstrated. Apart from the highly nonlinear objects we experiment with, we provide several experimental indicators that our proposed method achieves this linearization in challenging cases.

An additional differentiating factor from the aforementioned works is that, to the best of our knowledge, this three-

stage approach has not been used in the past for a related task.

2.1. cGAN and Adversarial Autoencoder

Let us briefly describe the two methods that consist our workhorse for learning the transformations. These are the conditional GAN and the Adversarial Autoencoder.

A Generative Adversarial Network (GAN) [5] is a generative network that has been very successfully employed for learning probability distributions [14]. A GAN is comprised of a generator G and a discriminator D network, where the generator samples from a pre-defined distribution in order to approximate the probability distribution of the training data, while the discriminator tries to distinguish between the samples originating from the model distribution to those from the data distribution. **Conditional GAN (cGAN)** [20] extends the formulation by conditioning the distributions with additional labels. More formally, if we denote with p_d the true distribution of the data, with p_z the distribution of the noise, with s the conditioning label and y the data, then the objective function is:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{s, y \sim p_d(s, y)} [\log D(s, y)] + \mathbb{E}_{s \sim p_d(s), z \sim p_z(z)} [\log(1 - D(s, G(s, z)))] \quad (1)$$

This objective function is optimized in an iterative manner, as

$$\min_{w_G} \max_{w_D} \mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{s, y \sim p_d(s, y)} [\log D(s, y; w_D)] + \mathbb{E}_{s \sim p_d(s), z \sim p_z(z)} [\log(1 - D(s, G(s, z; w_G); w_D))]$$

where w_G, w_D denote the generator’s, discriminator’s parameters respectively.

An Autoencoder (AE) [9, 19] is a neural network with two parts (an encoder and a decoder) and aims to learn a latent representation z of their input y . Autoencoders are mostly used in an unsupervised learning context [12] with the loss being the reconstruction error. On the other hand, an **Adversarial Autoencoder (AAE)** [18] consists of two sub-networks: i) a generator (an AE network), ii) a discriminator. The discriminator, which is motivated by GAN’s discriminator, accepts the latent vector (generated by the encoder) and tries to match the latent space representation with a pre-defined distribution.

3. Method

The core idea of our approach consists in finding a low-dimensional space that is (approximately) linear with respect to the projected representations. We aim to learn the (forward and inverse) transformations from the image space to the low-dimensional space. We know that an image $i^{(t)}$ is an instance of a dynamic scene at time t , hence the difference between the representations of two temporally close

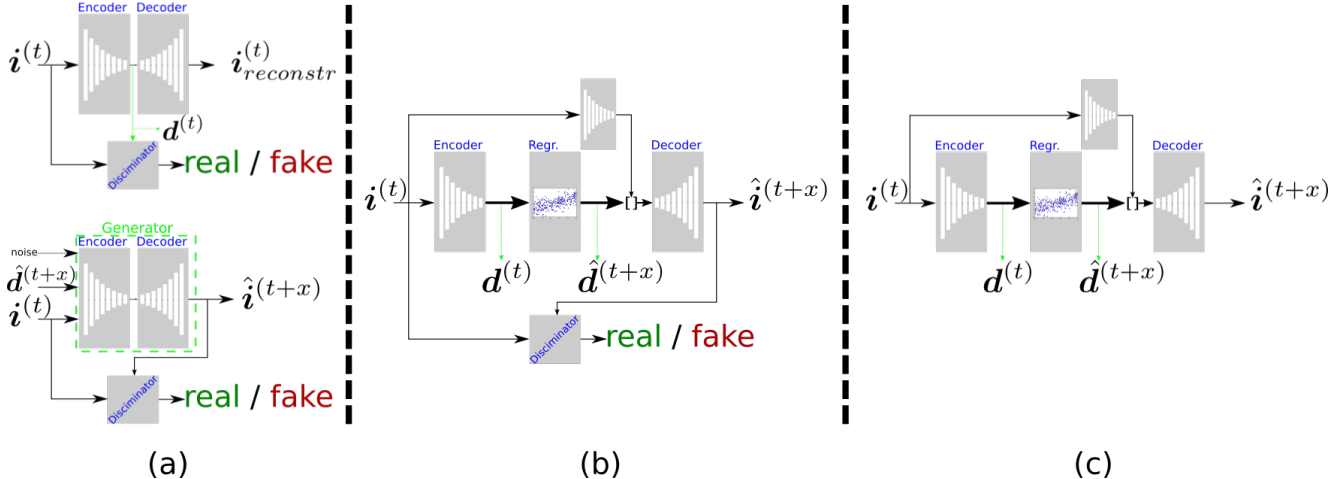


Figure 3: The architectures used in (a) separate training per step (the network for Stage I is on the top, for Stage III on the bottom), (b) fine-tuning of the unified model, (c) prediction. The ‘[]’ symbol denotes concatenation.

moments should be small and linear. We can learn the linear transitions of the representations and transform our image to $i^{(t+x)}$. We perform this linearization in 2-steps; an additional step is used to synthesize images of the same object with slightly different representations. The synthesized image can be thought of as a locally transformed image, e.g. the scene at $t+x$ moment with x sufficiently small.

3.1. Stage I: Latent image representation

Our goal consists in learning the transformations to the linearized space, however for the majority of the objects there are not enough videos annotated that can express a sufficient percent of the variation. For instance, it is not straightforward to find long videos of all breeds of dogs where the full body is visible. However, there are far more static images available online, which are faster to collect and can be used to learn the transformation from the image space to the latent space.

In an unsupervised setting a single image $i^{(t)}$ (per step) suffices for learning latent representations, no additional labels are required, which is precisely the task that Autoencoders were designed for. The latent vector of the Autoencoder lies in the latent space we want to find.

We experimentally noticed that the optimization converged faster if we used an adversarial learning procedure. We chose an Adversarial Autoencoder (AAE) [18] with a customized loss function. The encoder f_e^I accepts an image $i^{(t)}$, encodes it to $d^{(t)}$; the decoder f_d^I reconstructs $i^{(t)}$. We modify the discriminator to accept both the latent representation and the reconstructed image as input (fake example) and try to distinguish those from the distribution sample and the input image respectively. Moreover, we add a loss term that captures the reconstruction loss, which in our case consists of i) an ℓ_1 norm and ii) ℓ_1 in the image gradients. Con-

sequently, the final loss function is comprised of the following two terms: i) the adversarial loss, ii) the reconstruction loss or:

$$\mathcal{L}^I = \mathcal{L}_{adver} + \lambda_I \mathcal{L}_{rec}^I \quad (2)$$

with

$$\mathcal{L}_{rec}^I = \|f_d^I(f_e^I(\mathbf{y})) - \mathbf{y}\|_{\ell_1} + \|\nabla f_d^I(f_e^I(\mathbf{y})) - \nabla \mathbf{y}\|_{\ell_1} \quad (3)$$

The vector \mathbf{y} in this case is a training sample $i_k^{(t_k)}$, while λ_I is a hyper-parameter.

3.2. Stage II: Linear Model Learning

In this stage the latent representation $d^{(t)}$ of an image $i^{(t)}$ (as learned from stage I) is used to learn a mapping to the latent representation $d^{(t+x)}$ of the image $i^{(t+x)}$; the simple method of linear regression is chosen as a very simple transformation we can perform in a linear space. Given N pairs of images² $\{(i_1^{(t_1)}, i_1^{(t_1+x_1)}), (i_2^{(t_2)}, i_2^{(t_2+x_2)}), \dots, (i_N^{(t_N)}, i_N^{(t_N+x_N)})\}$, the set of the respective latent representations $\mathcal{D} = \{(d_1^{(t_1)}, d_1^{(t_1+x_1)}), (d_2^{(t_2)}, d_2^{(t_2+x_2)}), \dots, (d_N^{(t_N)}, d_N^{(t_N+x_N)})\}$; the set \mathcal{D} is used to learn the linear mapping:

$$d^{(t_j+x_j)} = \mathbf{A} \cdot [d^{(t_j)}; 1] + \epsilon \quad (4)$$

where ϵ is the noise; the Frobenius norm of the residual consists the error term:

$$L = \|d^{(t_j+x_j)} - \mathbf{A} \cdot [d^{(t_j)}; 1]\|_F^2 \quad (5)$$

To ensure the stability of the linear transformation we add a Tikhonov regularization term (i.e, Frobenius norm)

²Each pair includes two highly correlated images, i.e. two nearby frames from a video sequence.

on Eq. 5. That is,

$$\mathcal{L}^{II} = \|\mathbf{d}^{(t_j+x_j)} - \mathbf{A} \cdot [\mathbf{d}^{(t_j)}; 1]\|_F^2 + \lambda_{II} \|\mathbf{A}\|_F^2, \quad (6)$$

with λ_{II} a regularization hyper-parameter. The closed-form solution to Eq. 6 is

$$\mathbf{A} = \mathbf{Y} \cdot \mathbf{X}^T \cdot (\mathbf{X} \cdot \mathbf{X}^T + \lambda_{II} \cdot \mathcal{I})^{-1}, \quad (7)$$

where \mathcal{I} denotes an identity matrix, \mathbf{X} , \mathbf{Y} two matrices that contain column-wise the initial and target representations respectively, i.e. for the k^{th} sample $\mathbf{X}(:, k) = [\mathbf{d}_k^{(t_k)}; 1]$, $\mathbf{Y}(:, k) = \mathbf{d}_k^{(t_k+x_k)}$.

3.3. Stage III: Latent representation to image

In this step, we want to learn a transformation from the latent space to the image space, i.e. the inverse transformation of Stage I. In particular, we aim to map the regressed representation $\hat{\mathbf{d}}^{(t+x)}$ to the image $\mathbf{i}^{(t+x)}$. Our prior distribution consists of a low-dimensional space, which we want to map to a high-dimensional space; GANs have experimentally proven very effective in such mappings [14, 25].

A conditional GAN is employed for this step; we condition GAN in both the (regressed) latent representation $\hat{\mathbf{d}}^{(t+x)}$ and the original image $\mathbf{i}^{(t)}$. Conditioning on the original image has experimentally resulted in faster convergence and it might be a significant feature in case of limited amount of training samples. Inspired by the work of [11], we form the generator as an autoencoder denoting the encoder as f_e^{III} , the decoder as f_d^{III} . Skip connections are added from the second and fourth layers of the encoder to the respective layers in the decoder with the purpose of allowing the low-level features of the original images to be propagated to the result.

In conjunction with [11] and Sec. 3.1, we add a reconstruction loss term as

$$\mathcal{L}_{rec}^{III} = \|f_d^{III}(f_e^{III}(\mathbf{y})) - \mathbf{s}\|_{\ell_1} + \|\nabla f_d^{III}(f_e^{III}(\mathbf{y})) - \nabla \mathbf{s}\|_{\ell_1} \quad (8)$$

where \mathbf{y} is a training sample $\mathbf{i}_k^{(t_k)}$ and \mathbf{s} is the conditioning label (original image) $\mathbf{i}_{k-x}^{(t_k-x)}$. In addition, we add a loss term that encourages the features of the real/fake samples to be similar. Those features are extracted from the penultimate layer of the AAE’s discriminator. Effectively, this leads the fake (i.e. synthesized) images to have representations that are close to the original image. The final objective function for this step includes three terms, i.e. the adversarial, the reconstruction and the feature loss:

$$\mathcal{L}^{III} = \mathcal{L}_{cGAN} + \lambda_{III} \mathcal{L}_{rec}^{III} + \lambda_{III,feat} \mathcal{L}_{feat} \quad (9)$$

where \mathcal{L}_{cGAN} is defined in Eq.1, \mathcal{L}_{feat} represents the similarity cost imposed on the features from the discriminator’s penultimate layer and λ_{III} , $\lambda_{III,feat}$ are scalar hyper-parameters. To reduce the amount of hyper-parameters in our work, we have set $\lambda_{III} = \lambda_I$.

3.4. End-to-end fine-tuning

Even though the training in each of the aforementioned three stages is performed separately, all the components are differentiable with respect to their parameters. Hence, Stochastic Gradient Descent (SGD) can be used to fine-tune the pipeline.

Not all of the components are required for the fine-tuning, for instance the discriminator of the Adversarial Autoencoder is redundant. From the network in Stage I, only the encoder is utilized for extracting the latent representations, then linear regression (learned matrix \mathbf{A}) can be thought of as a linear fully-connected layer. From network in Stage III, all the components are kept. The overall architecture for fine-tuning is depicted in Fig. 3.

3.5. Prediction

The structure of our three-stage pipeline is simplified for performing predictions. The image $\mathbf{i}^{(t)}$ is encoded (only the encoder of the network in Stage I is required); the resulting representation $\mathbf{d}^{(t)}$ is multiplied by \mathbf{A} to obtain $\hat{\mathbf{d}}^{(t+x)}$, which is fed into the conditional GAN to synthesize a new image $\hat{\mathbf{i}}^{(t+x)}$. This procedure is visually illustrated in Fig. 3, while more formally:

$$\hat{\mathbf{i}}^{(t+x)} = f_d^{III}(f_e^{III}(\mathbf{A} \cdot [f_e^I(\mathbf{i}^{(t)}; 1)], \mathbf{i}^{(t)})) \quad (10)$$

3.6. Network architectures

Our method includes two networks, i.e. an Adversarial Autoencoder for Stage I and a conditional GAN for Stage III. The encoder/decoder of both networks share the same architecture, i.e. 8 convolutional layers followed by batch normalization [10] and LeakyRELU [16]. The discriminator consists of 5 layers in both cases, while the dimensionality of the latent space is 1024 for all cases. Please refer to the table in the supplementary material for further details about the layers.

4. Experiments

In this Section we provide the details of the training procedure along with the dedicated qualitative and quantitative results for all three objects, i.e. human faces, cats’ faces and dogs’ faces. Our objective is to demonstrate that this augmentation leads to learning invariances, e.g. deformations, not covered by commonly used techniques.

4.1. Implementation details

The pairs of images required by the second and third stages, were obtained by sequential frames of that object. Different sampling of x was allowed per frame to increase the variation. To avoid the abrupt changes between pairs

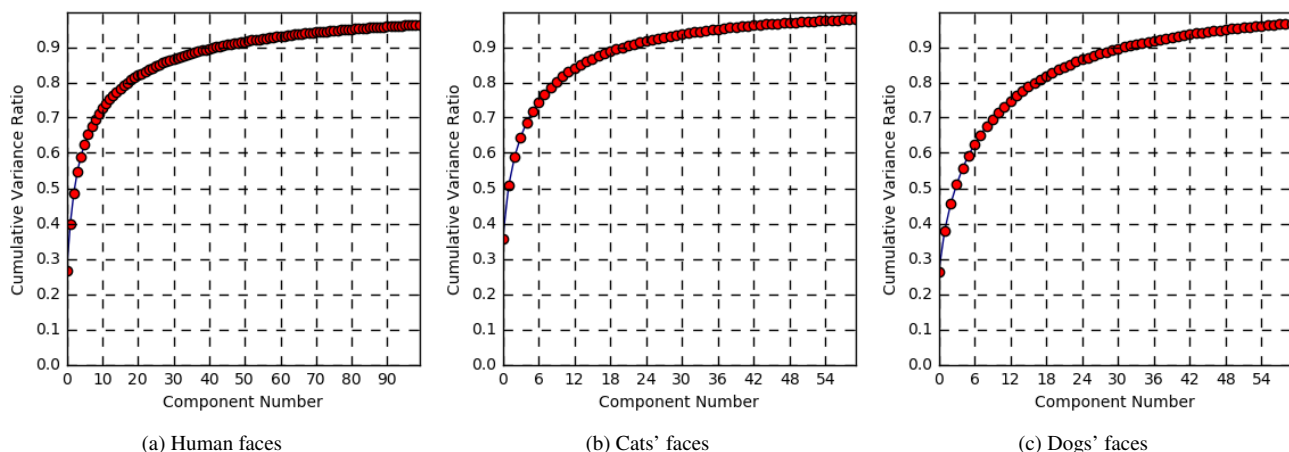


Figure 4: Average variance in the dynamics representation per video for (a) the case of human faces, (b) cats' faces, (c) dogs' faces.



Figure 5: (Preferably viewed in color) Conditional, iterative prediction from our proposed method. The images on the left are the original ones; then from the left to the right the i^{th} column depicts the $(i - 1)^{th}$ synthesized image (iteration $(i - 1)$). In both rows, the image on the left is animated, hence if opened with Adobe Acrobat reader the transitions will be auto-played.

of frames, the structural similarity (SSIM) of a pair was required to lie in an interval, i.e. the frames with i) zero, ii) excessive movement were omitted.

Each of the aforementioned stages was trained separately; after training all of them, we have performed fine-tuning in the combined model (all stages consist of convolutions). However, as is visually illustrated in figures in the supplementary material there are minor differences in the two models. The results of the fine-tuned model are marginally more photo-realistic, which consists fine-tuning optional.

4.2. Datasets

A brief description of the databases utilized for training is provided below:

Human faces: The recent dataset of MS Celeb [7] was employed for Stage I (Sec. 3.1). MS Celeb includes 8,5 million facial images of 100 thousand celebrities consisting it one of the largest public datasets for static facial images. In our case, the grayscale images were excluded, while from the remaining images a subset of 2 million random images was sampled. For the following two stages that require pairs of images the dataset of 300 Videos in-the-wild (300VW) [28] was employed. This dataset includes 114 videos with approximately 1 minute duration each. The to-

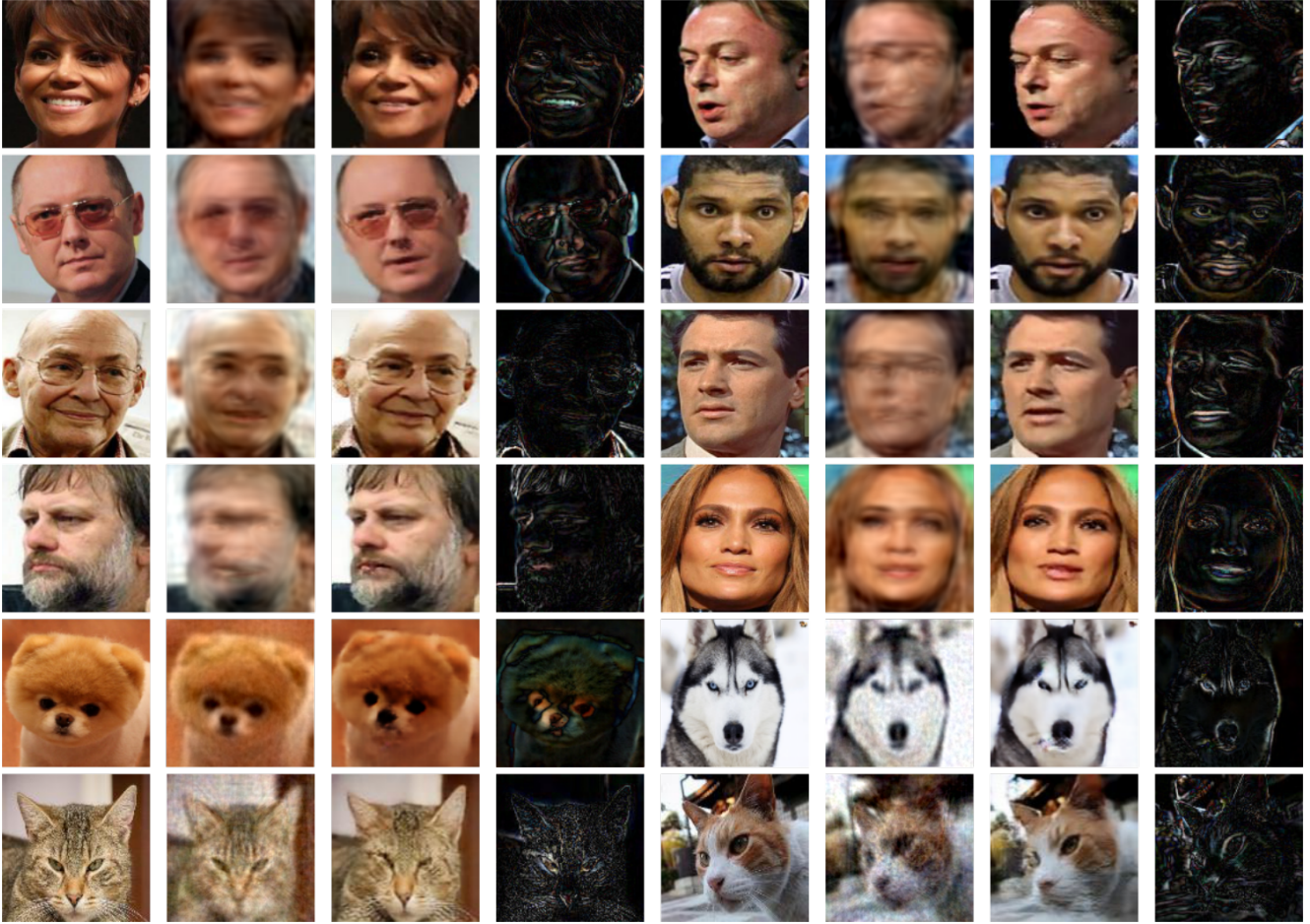


Figure 6: (Preferably viewed in color) Visual results of the synthesized images. There are four columns from the left to the right (split into left and right parts) which depict: (a) the original image, (b) the linear model (PCA + regression), (c) our proposed method, (d) the difference in intensities between the proposed method and the original image. The difference does not depict accurately the pose variation; the gif images in the supplementary material demonstrate the animated movement. Nevertheless, some noticeable changes are the following: a) in the left part in the second, and fifth images there is a considerable 3D rotation (pose variation), b) in the first, third and sixth in the left split there are several deformations (eyes closing, mouth opening etc.), c) in the second image on the right part, the person has moved towards the camera.

tal amount of frames sampled for Stage II (Sec. 3.2) is 13 thousand frames; 10 thousand frames are sampled for validation, while the rest are used for training the network in Stage III (Sec. 3.3).

Cat faces: The pet dataset of [24] was employed for learning representations of cats’ faces. The dataset includes 37 different breeds of cats and dogs (12 for cats) with approximately 200 images each³. In addition to those, we collected 1000 additional images, for a total of 2000 images. For the subsequent stages of our pipeline, pairs of images were required, hence we have collected 20 videos with an average duration of 200 frames. The head was detected with the DPM detector of [4] in the first frame and the rest

³Each image is annotated with a head bounding box.

tracked with the robust MDNET tracker of [22]. Since the images of cats are limited, the prior weights learned for the (human) facial experiment were employed (effectively the pre-trained model includes a prior which we adapt for cats).

Dog faces: The Stanford dog dataset [13] includes 20 thousand images of dogs from 120 breeds. The annotations are in the body-level, hence the DPM detector was utilized to detect a bounding box of the head. The detected images, i.e. 8 thousand, consisted the input for Stage I of our pipeline. Similarly to the procedure for cats, 30 videos (with average duration of 200 frames) were collected and tracked for Stages II and III.

4.3. Variance in the latent space

A quantitative self-evaluation experiment was to measure the variance of latent representations per video. The latent representations of sequential frames should be highly correlated; hence the variance in a video containing the same object should be low.

A PCA was learned per video and the cumulative eigenvalue ratio was computed. We repeated the same procedure for all the videos (per object) and then averaged the results. The resulting plots with the average cumulative ratio are visualized in Fig. 4. In the videos of the cats and the dogs, we observe that the first 30 components express 90% of the variance. In the facial videos that are longer (over 1500 frames) the variance is greater, however the first 50 components explain over 90% of the variance.

4.4. Qualitative assessment

Considering the sub-space defined by PCA as the latent space and learning a linear regression there is the linear counterpart of our proposed method. To demonstrate the complexity of the task, we have learned a PCA per object⁴; the representations of each pair were extracted, linear regression was performed and then the regressed representations were used to create the new sample.

In Fig. 6, we have visualized some results for all three cases (human, cats' and dogs' faces). In all cases the images were not seen during the training with the cats' and dogs' images being downloaded from the web (all were recently uploaded), while the faces are from WIKI-DB dataset [27]. The visualizations verify our claims that a linear transformation in the latent space, can produce a realistic non-linear transformation in the image domain. In all of the facial images there is a deformation of the mouth, while in the majority of them there is a 3D movement. On the contrary, on the dogs' and the cats' faces, the major source of deformation seems to be the 3D rotation. An additional remark is that the linear model, i.e. regressing the components of PCA, does not result in realistic new images, which can be attributed to the linear assumptions of PCA.

Aside of the visual assessment of the synthesized images, we have considered whether the new synthesized image is realistic enough to be considered as input itself to the pipeline. Hence, we have run an iterative procedure of applying our method, i.e. the outcome of iteration k becomes the input to iteration $k + 1$. Such an iterative procedure essentially creates a collection of different images (constrained to include the same object of interest but with slightly different latent representations). Two such collections are depicted in Fig. 5, where the person in the first row performs a 3D movement, while in the second different

⁴To provide a fair comparison PCA received the same input as our method (i.e. there was no effort to provide further (geometric) details about the image, the pixel values are the only input).

deformations of the mouth are observed. The image on the left is animated, hence if opened with Adobe Acrobat reader the transitions will be auto-played. We strongly encourage the reviewers to view the animated images and check the supplementary animations.

4.5. Age estimation with augmented data

To ensure that a) our method did not reproduce the input to the output, b) the images are close enough (small change in the representations) we have validated our method by performing age estimation with the augmented data.

We utilized as a testbed the AgeDB dataset of [21], which includes 16 thousand manually selected images. As the authors of [21] report, the annotations of AgeDB are accurate to the year, unlike the semi-automatic IMDB-WIKI dataset of [27]. For the aforementioned reasons, we selected AgeDB to perform age estimation with i) the original data, ii) the original plus the new synthesized samples. The first 80% of the images was used as training set and the rest as test-set. We augmented only the training set images with our method by generating one new image for every original one. We discarded the examples that have a structural similarity (SSIM) [31] of less than 0.4 with the original image; this resulted in synthesizing 6 thousand new frames (approximately 50% augmentation).

We trained a Resnet-50 [8] with i) the original training images, ii) the augmented images and report here the Mean Absolute Error (MAE). The pre-trained DEX [27] resulted in a MAE of 12.8 years in our test subset [21], the Resnet with the original data in MAE of 11.4 years, while with the augmented data resulted in a MAE of 10.3 years, which is a 9.5% relative decrease in the MAE. That dictates that our proposed method can generate new samples that are not trivially replicated by affine transformations.

5. Conclusion

In this work, we have introduced a method that finds a low-dimensional (approximately) linear space. We have introduced a three-stage approach that learns the transformations from the highly non-linear image space to the latent space along with the inverse transformation. This approach enables us to make linear changes in the space of representations and these result in non-linear changes in the image space. The first transformation was approximated by an Adversarial Autoencoder, while a conditional GAN was employed for learning the inverse transformation and acquiring the synthesized image. The middle step consists of a simple linear regression to transform the representations. We have visually illustrated that i) the representations of a video form a discrete cluster (T-SNE in Fig. 2) ii) the representations of a single video are highly correlated (average cumulative eigenvalue ratio for all videos).

References

- [1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. [3](#)
- [2] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999. [1](#), [2](#)
- [3] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CVPR*, 2017. [1](#)
- [4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *T-PAMI*, 32(9):1627–1645, 2010. [7](#)
- [5] 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. [3](#)
- [6] R. Goroshin, M. F. Mathieu, and Y. LeCun. Learning to linearize under uncertainty. In *NIPS*, pages 1234–1242, 2015. [3](#)
- [7] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *ECCV*, pages 87–102, 2016. [6](#)
- [8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*. IEEE, 2016. [1](#), [8](#)
- [9] G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In *NIPS*, 1994. [3](#)
- [10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. [5](#)
- [11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. [3](#), [5](#)
- [12] M. Kan, S. Shan, H. Chang, and X. Chen. Stacked progressive auto-encoders (spae) for face recognition across poses. In *CVPR*, 2014. [3](#)
- [13] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshops*, volume 2, page 1, 2011. [7](#)
- [14] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *CVPR*, 2017. [3](#), [5](#)
- [15] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *ICLR*, 2017. [3](#)
- [16] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. [5](#)
- [17] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008. [2](#)
- [18] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. [3](#), [4](#)
- [19] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *ICANN*, 2011. [3](#)
- [20] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. [3](#)
- [21] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *CVPR Workshops*, 2017. [8](#)
- [22] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. [7](#)
- [23] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016. [3](#)
- [24] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and dogs. In *CVPR*, 2012. [7](#)
- [25] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, pages 2536–2544, 2016. [3](#), [5](#)
- [26] K. Rematas, T. Ritschel, M. Fritz, and T. Tuytelaars. Image-based synthesis and re-synthesis of viewpoints guided by 3d models. In *CVPR*, 2014. [1](#)
- [27] R. Rothe, R. Timofte, and L. Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, pages 1–14, 2016. [8](#)
- [28] J. Shen, S. Zafeiriou, G. Chrysos, J. Kossai, G. Tzimiropoulos, and M. Pantic. The first facial landmark tracking in-the-wild challenge: Benchmark and results. In *300-VW in ICCV-W*, December 2015. [2](#), [7](#)
- [29] L. Tran, X. Yin, and X. Liu. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, volume 4, page 7, 2017. [1](#), [2](#)
- [30] C. Vondrick, H. Pirsaviash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*, pages 613–621, 2016. [3](#)
- [31] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 13(4):600–612, 2004. [8](#)
- [32] Y. Wu, T. Hassner, K. Kim, G. Medioni, and P. Natarajan. Facial landmark detection with tweaked convolutional neural networks. *arXiv preprint arXiv:1511.04031*, 2015. [2](#)
- [33] T. Xue, J. Wu, K. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NIPS*, pages 91–99, 2016. [3](#)
- [34] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision*, pages 517–532. Springer, 2016. [3](#)
- [35] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016. [3](#)
- [36] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li. Face alignment across large poses: A 3d solution. In *CVPR*, pages 146–155, 2016. [2](#)
- [37] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017. [1](#)