HAL
open science

Bilinear image translation for temporal analysis of photo collections

Théophile Dalens, Mathieu Aubry, Josef Sivic

To cite this version:

Théophile Dalens, Mathieu Aubry, Josef Sivic. Bilinear image translation for temporal analysis of photo collections. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, 10.1109/TPAMI.2019.2950317 . hal-02452091

HAL Id: hal-02452091

https://inria.hal.science/hal-02452091

Submitted on 23 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Bilinear image translation for temporal analysis of photo collections

Théophile Dalens, Mathieu Aubry and Josef Sivic

Abstract—We propose an approach for analyzing unpaired visual data annotated with time stamps by generating how images would have looked like if they were from different times. To isolate and transfer time dependent appearance variations, we introduce a new trainable bilinear factor separation module. We analyze its relation to classical factored representations [1] and concatenation-based auto-encoders [2]. We demonstrate this new module has clear advantages compared to standard concatenation when used in a bottleneck encoder-decoder convolutional neural network architecture. We also show that it can be inserted in a recent adversarial image translation architecture [3], enabling the image transformation to multiple different target time periods using a single network. We apply our model to a challenging collection of more than 13,000 cars manufactured between 1920 and 2000 [4] and a dataset of high school yearbook portraits from 1930 to 2009 [5]. This allows us, for a given new input image, to generate a "history-lapse video" revealing changes over time by simply varying the target year. We show that by analyzing the generated history-lapse videos we can identify object deformations across time, extracting interesting changes in visual style over decades.

1 Introduction

Large-scale quantitative analysis of temporal data has made a great impact in science. For example, in the text domain, word frequency analysis of millions of books made it possible to study cultural trends quantitatively [6]. In the speech domain, analysis of years of speech data made it possible to study child's acquisition of language [7]. Similarly, leveraging large-scale time stamped *visual data* would enable new applications such as assessing disease progression in radiology [8], [9] or discovering new insights about long-term evolution of visual style in art, history, architecture or design [5], [10], [11], [12]. However, similar large-scale analysis of *visual data* is a notoriously difficult task because of two fundamental challenges [4], [13], [14], [15]. First, we need to define the appropriate visual vocabulary. For example, what is the right visual vocabulary to describe changes in style of cars between 1960s and 1970s [4]? While the vocabulary is rather well defined in the text domain, the visual vocabulary of image fragments is very large, a priori unknown and depends on the task. Second, the input temporal image collections typically do not contain paired data, i.e. it is hard to observe the evolution of one object over (long) periods of time. Hence, it is necessary to extract temporal variations from different object instances observed at different times.

Previous work on visual discovery has addressed the above issues by finding a vocabulary of characteristic parts (such as windows, doors or car lights) [13], [14] and their correspondences across time [4], [15] via training discriminative part detectors based on Histogram of Oriented Gradients (HOG) [16]. However, such representation provides only a coarse sampling of the image limited by the discrete set of locations of the discriminative parts. Detailed pixel-level analysis of visual data that would enable visual discovery of subtle visual patterns still remains a difficult problem. To address this issue, we develop a model based on a convolutional neural network (CNN) architecture that operates directly on the pixel level, i.e. takes an image as input and translates it into an output image similar to the input, but modifying its appearance so that it looks similar to images from a different given target time period. The output of this model

Fig. 1. **Bilinear image translation.** Our method takes as input an image of an object (in green), such as a car, and generates what it would have looked like in another time-period (in blue). Each row shows temporal translation for a different input car image (in green). The translation model is trained on a unpaired dataset of cars with time stamps. We show that analyzing changes between the generated images reveal structural deformations in car shape and appearance over time.

- *T. Dalens and J. Sivic are with Inria and the Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France.*
  E-mail: {theophile.dalens, josef.sivic}@inria.fr
- *J. Sivic is also with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague.*
- *M. Aubry is with Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-77455 Marne-la-Vallée, France.*
  E-mail: mathieu.aubry@imagine.enpc.fr

for the same input image but different target time periods can be used to identify and visualize changes over time that are relevant for the specific input image. As a result, we can make videos of how a particular input car would look like if it was made in different decades, highlight design elements, such as running boards, that appear and disappear over time, or even measure deformations over time by computing displacement flow. We analyze how existing architectures can be used for this task, show their limitations, introduce a new bilinear factorization module and demonstrate its advantages. This module implicitly learns both the (i) visual vocabulary for the time translation task and (ii) correspondences across input object instances. The outcome is a model that transfers time specific appearance variation, learnt from the data, to a given input object instance generating its unseen appearance across time, as illustrated in figure 1.

Contributions. Our contributions are three-fold. First, we introduce *image translation as a tool for visual discovery* in unpaired historical image collections and demonstrate visual discovery results on two challenging datasets. Second, we introduce a new *bilinear factorization module*, relate it to existing approaches and show its advantages compared to standard concatenation-based factor representation when used in a bottleneck auto-encoder architecture. Third, we show that our bilinear module can be plugged into a modern unpaired image translation architecture [3], leading to state-of-the-art results with clear computational advantages.

Roadmap. The rest of the paper is organized as follows. After reviewing related work in section 2, we describe in section 3 the idea of visual discovery by learning a factored image representation. The details of our new bilinear module, and its relation to existing approaches, are given in section 4. We then describe in section 5 how this module can be inserted either in a bottleneck auto-encoder or in a modern image translation architecture. Finally, we present a quantitative evaluation of our method, and visual discovery results in section 6.

Source code and additional results can be found on the probject webpage, https://www.di.ens.fr/willow/research/bilineartranslation.

2 Related work

We bring together three separate lines of work: (i) visual discovery [4], [5], [13], [14], [15], (ii) image representations with multi-linear models [1], [17], [18], [19], [20] and (iii) generative convolutional neural networks [2], [21], [22], [23], [24], which we review below.

2.1 Visual discovery

Visual mining and temporal analysis has been successfully addressed for object and scene instances [25], [26], [27], [28], but here we focus on the much harder problem of finding visual patterns that generalize beyond instance-level matching. Our work is related to learning from unlabelled data, including unsupervised algorithms for object categorization [29], [30], [31], [32], [33], [34] and segmentation [31], [35], [36]. However, being unsupervised, these models are limited to discovering only visual patterns that are both very common and highly visually consistent. To address that issue, *discriminative* discovery models [4], [13], [14], [15], [37] further constrain the problem by readily available, though only weak (image-level), supervision, which has enabled finding less common visual patterns, for example, architectural elements that distinguish one city from other cities [13] or cars manufactured in different decades [4], but also distinctive parts of objects [14], [37] and scenes [38]. Others have looked at identifying parts of images that are temporally discriminative in order to date historical objects in photographs [39]. This is done by passing partially occluded images through the network trained to estimate the date; the parts of the image that lead to the high discrepancy in neural activation of the higher layers of the network are deemed as discriminative. The limitation of all the above methods is that they reduce each object to only its most distinctive parts. In contrast, we develop a visual discovery model that learns a factored image representation to separate the visual content characteristic for the given target attribute but that is forced at the same time to reconstruct the entire image. This enables discovery of visual patterns that go beyond the sparse set of distinctive parts. Finally, while we focus on temporal analysis of appearance and shape of objects depicted in photographs. It is also possible to date a historical photograph based on the used color imaging process [40].

2.2 Image representation with (multi-)linear models

Since eigenfaces [17], [18] linear models, such as Principal Component Analysis (PCA) have been used to represent images. However, PCA is often not well suited to represent complex interactions between factors because of its simple linear additive nature, leading to other, more complex, forms of interactions. For example, Tenenbaum and Freeman [1] have explored multiplicative interactions in a bilinear model to factor out style and content for font and face analysis. More recently, Tensor Analyzers [20] have further explored multilinear interactions [19] with a Gaussian priors over the latent variables. These approaches, however, consider the latent variables as unknown, while we suppose that the time stamp of each image is known both at training and testing. Also, these models have rarely been used for more complex images than aligned faces, because the complex variations that occur in natural images are hard to represent using linear models in pixel space. In contrast, we formulate bilinear factor translation in a end-to-end trainable encoder-decoder CNN architecture. Bilinear layers have recently been used in CNNs, for example in the context of fine-grained recognition [41] and visual question answering [42], but to the best of our knowledge, they were not used for visual discovery by image translation.

2.3 Image translation with (factored) convolutional neural networks

Neural network based models, such as Boltzmann machines [43], [44], [45], [46], auto-encoders [47], or more recently variational auto-encoders [2] have been long used for image generation. Typically, parameters of the network

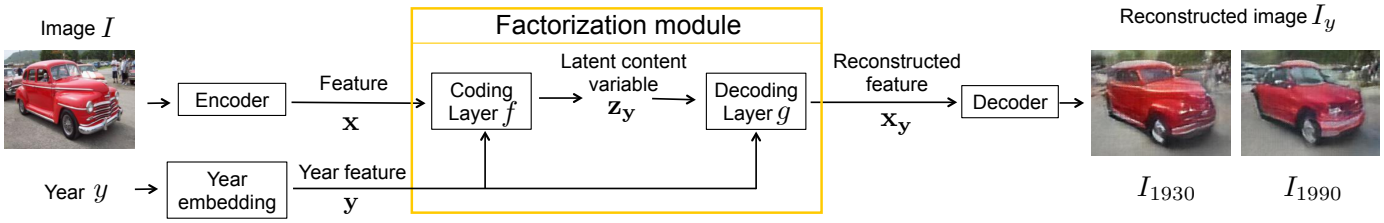


Fig. 2. **CNN architecture for factored visual discovery.** We assume that an image is completely determined by (i) latent content variable $\mathbf{z_y}$ capturing the appearance factors independent of time such as car type and color, and (ii) the time feature $\mathbf{y}$ capturing the appearance factors specific to certain time, such as the running boards for cars made in 1930s. The input image I is encoded into a feature vector $\mathbf{x}$, which is transformed together with year variable $\mathbf{y}$ using a mapping $f(\mathbf{x}, \mathbf{y})$ into a latent content representation $\mathbf{z_y}$. A second mapping $g(\mathbf{z}, \mathbf{y})$ then combines the latent content with the input time feature $\mathbf{y}$ to produce new feature $\mathbf{x_y}$. The reconstructed feature $\mathbf{x_y}$ is finally decoded into image I_y. Parameters of the factorization module together with the encoder and decoder are learnt in an end-to-end manner.

are learnt to minimize some form of *reconstruction cost* on the set of training images. To modify images, it is possible to learn a specific intermediate representation which aims to disentangle factors of variation in the observed visual data such as viewpoint, lighting [48], [49], [50], style of digits [23] or facial attributes [51], [52]. This is typically achieved by minimizing an additional *separation cost* to encourage independence between subsets of the hidden (or observed) variables. The existing network architectures for disentangling are based on an additive combination of factors. In contrast, we develop a bilinear factor module capable of modelling multiplicative interactions and demonstrate its benefits on the task of visual discovery in temporal data. However, until recently, these models were limited to simple controlled set-ups such as MNIST digits or small image patches.

As shown by [21], [22], [24], [53], [54], [55], the quality of the output imagery can be further enhanced using *adversarial costs* [56] that encourage the network to generate images that cannot be distinguished from natural images.

Especially related to ours is also the work of [57] that describes recurrent architecture to generate temporal object transformations (such as food degradation). However, the model assumes example temporal transformations are given at training time in the form of time-lapse videos and hence is not applicable in our set-up, where each object instance is observed at only one time. To generate complex photo-realistic without losing low-level details, it is necessary to modify the network architecture by adding shortcut connections. This was demonstrated for very different sets of paired images in [58], and later extended to unpaired data in [3]. However, in contrast to factored generation approaches, this requires training a separate network for each type of change, without learning a shared representation from all available data. This makes the approach of [3] impractical for generating continuous and, in particular, temporal changes. Our approach will allow to overcome this limitation. Recently, this limitation has been also addressed in [59] that also overcomes the need to train multiple models but uses different losses and does not directly address continuous attributes such as time.

3 Visual discovery by learning factored image representation

We seek to discover patterns in changes of appearance related to time, or more generally, a variable associated to images. For example, what are the visual features characteristic for the style of cars in 1960s? What makes the style different from cars made in 1940s? We propose to address these questions by generating images corresponding to "what the input image would have looked like at a different time", using a factored model that separates time-specific appearance variation from the other content of the image. This is illustrated in figure 2. This allows us, for a given input image, to generate a "history-lapse video" revealing changes over time by simply varying the input target time variable.

Factored representation. Our main hypothesis is that an image is completely determined by latent content variable $\mathbf{z_y} \in \mathbb{R}^C$ and time variable $\mathbf{y}$. For example, the appearance of a particular car from the 1930s is a combination of two sets of factors. The first set of factors, encoded in a content vector $\mathbf{z_y} \in \mathbb{R}^C$, is independent of the time and includes, for example, viewpoint and body type (e.g. sedan vs. pick-up truck). The second set of factors, determined by a time vector $\mathbf{y}$, is related to the period the car was built in, due to cars exhibiting time-characteristic features, such as running boards or round mud guards for cars from the 1930s.

Model description. As illustrated in figure 2, we assume each input image is associated with a time stamp, represented by a feature vector $\mathbf{y} \in \mathbb{R}^T$. We denote by $\mathbf{x} \in \mathbb{R}^K$ a K-dimensional feature encoding of image I. Our model computes a mapping of these features into a latent content factor $\mathbf{z_y} = f(\mathbf{x}, \mathbf{y})$. Then the inverse mapping $\mathbf{x_y} = g(\mathbf{z_y}, \mathbf{y})$ decodes the content factor $\mathbf{z_y}$ jointly with the time dependent factor $\mathbf{y}$ into a new image feature $\mathbf{x_y}$ that is reconstructed into the output image $I_\mathbf{y}$. The specific form of f and g is discussed in detail section 4.

Training. The model is trained in an end-to-end manner from a dataset of images with time stamps. We assume the dataset is *unpaired*, i.e. each object is observed at only one time. For example, we observe a particular car made at one specific time period and we do not know how that particular car would have looked like if it was made at a different time period. The objective of the training is to isolate the time dependent appearance variation from such unpaired input training data. This is achieved by optimizing a cost function that includes a reconstruction term and an optional translation term. The reconstruction term ensures that the model reconstructs well the input training images at their given time period. The translation term ensures that

when the input image is translated to different time periods, the output images exhibit characteristics of the training images from those target periods. As we don't have paired training data, this is implemented in an adversarial manner, i.e. the translated images should be indistinguishable by a classifier from the training images in the specific target time periods.

Temporal discovery. Interestingly, an image feature $\mathbf{x}$, associated in the dataset with a time feature $\mathbf{y}$, can be translated to a different time by simple changing the value of the time feature $\mathbf{y}'$

$$\mathbf{x}_{\mathbf{y}'} = g\left(f\left(\mathbf{x}, \mathbf{y}'\right), \mathbf{y}'\right) , \tag{1}$$

which can be decoded to output image $M(I, y')$. We can then compare the $M(I, y')$ for different times y' to the original image I, and identify what in the image would have been different had it been associated to a different time stamp. Such visual temporal analysis is presented in section 6.4.

4 Bilinear factor separation module

In this section we describe our new bilinear factored representation module, which is visualized in figure 3a, and that can be used as the core factorization module in the visual discovery architecture shown in figure 2. It takes as input the feature $\mathbf{x}$ of an image and the year $\mathbf{y}$, and returns a feature $\mathbf{x}_\mathbf{y}$ corresponding to what the feature would have been if its time had been defined by $\mathbf{y}$.

4.1 Module description

The key idea of our factored translation approach is to combine the content vector $\mathbf{z} \in \mathbb{R}^S$ and the style vector $\mathbf{y} \in \mathbb{R}^T$ in a multiplicative way, using a bilinear layer. In particular, the layer $g(\mathbf{z}_\mathbf{y}, \mathbf{y})$ reconstructs the feature $\mathbf{x}_\mathbf{y}$ from the latent content variable $\mathbf{z}_\mathbf{y}$ and the given input time variable $\mathbf{y}$ as

$$\mathbf{x}_\mathbf{y}^k = g(\mathbf{z}_\mathbf{y}, \mathbf{y})^k = \sum_{s=1}^{S} \sum_{t=1}^{T} W_g^{s,t,k} \mathbf{z}_\mathbf{y}^s \mathbf{y}^t , \tag{2}$$

where W_g are the learnable weights of the layer g and a^b denotes component b of the vector a (and similarly for tensor components). Note that if $\mathbf{y}$ is a one hot encoding of the time period y, matrices $W_g^{\cdot,t,\cdot}$ can be interpreted as linear weights specific to the period y.

Before we can proceed with translation using equation (2), the latent content vector $\mathbf{z}_\mathbf{y}$ needs to be estimated from the input image feature $\mathbf{x}$. We compute $\mathbf{z}_\mathbf{y}$ using another bilinear layer as

$$\mathbf{z}_\mathbf{y}^s = f(\mathbf{x}, \mathbf{y})^s = \sum_{k=1}^{K} \sum_{t=1}^{T} W_f^{k,t,s} \mathbf{x}^k \mathbf{y}^t, \tag{3}$$

where W_f are the weights of the layer f and the $\mathbf{z}_\mathbf{y}^s$ is the component s of the latent content variable $\mathbf{z}_\mathbf{y}$ representing the input $\mathbf{x}$ interpreted using the given input time vector $\mathbf{y}$. Note also that similarly to W_g, if $\mathbf{y}$ is a one hot encoding of the time period y, matrices $W_f^{\cdot,t,\cdot}$ can be interpreted as weights of a linear layer for time period t. Note that the latent content variable $\mathbf{z}_\mathbf{y}$ depends on the given input time vector $\mathbf{y}$ and can be thought of as a "projection" of the input representation $\mathbf{x}$ onto an "image subspace" specific to period y.

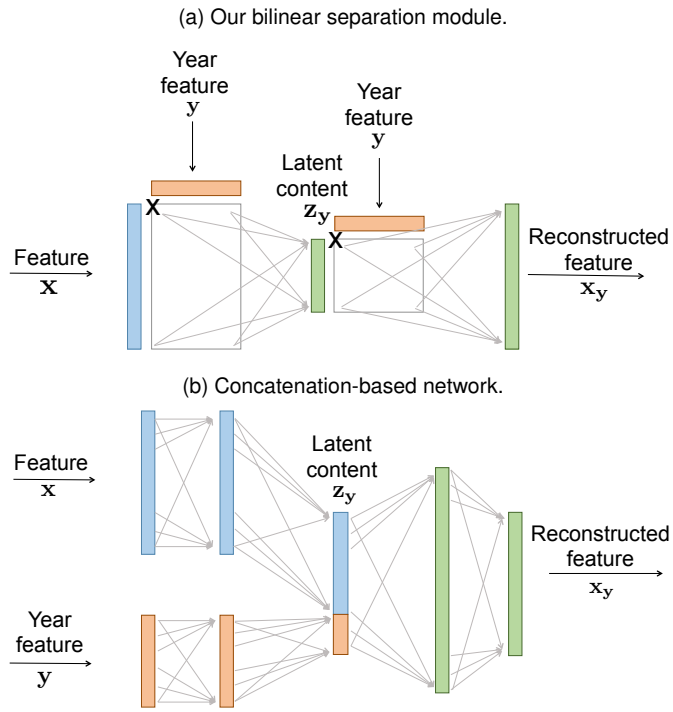


Fig. 3. **Factorization architectures.** Our bilinear factorization module (top) and the standard concatenation-based architecture (bottom) have two principal differences. First, our module captures multiplicative interactions between time $\mathbf{y}$ and content $\mathbf{z}_\mathbf{y}$, while concatenation implies additive interactions. Then, we explicitly include dependency on time $\mathbf{y}$ in computing latent content $\mathbf{z}_\mathbf{y}$. More layers can be included in our bilinear module similar to the concatenation architecture.

The resulting architecture is shown in figure 3a. Our bilinear factor separation module is composed of two bilinear layers. The first bilinear layer $f(\mathbf{x}, \mathbf{y})$, given by eq. (3), combines the input feature $\mathbf{x}$ and the given input year $\mathbf{y}$ in a multiplicative fashion into a content vector $\mathbf{z}_\mathbf{y}$. The second bilinear layer, given by eq. (2), combines the content vector $\mathbf{z}_\mathbf{y}$ with the input time $\mathbf{y}$ in a multiplicative way into the reconstructed feature $\mathbf{x}_\mathbf{y}$.

The presence of the bilinear module is crucial for architectures that have a bottleneck. When there is a bottleneck, the goal of our model is to store in the bottleneck code $\mathbf{z}_\mathbf{y}$ only the *time independent* information (e.g. the type and color of the car) and not the appearance specific to the period y (e.g. the shape of mudguards specific to the time period). This time-specific information thus needs to be added back by combining $\mathbf{z}_\mathbf{y}$ with y to produce the reconstructed feature $\mathbf{x}_\mathbf{y}$ as illustrated in figure 2.

4.2 Relation to other approaches

In the following we discuss the relation of our bilinear module to (i) principal component analysis, (ii) style and content factorziation and (iii) concatenation-based image translation.

4.2.1 *Relation to principal component analysis (PCA)*

In the case where $\mathbf{y}$ is a one hot vector representing each time period, there is a direct relation between our bilinear

module and PCA. Our module can, in this case, be rewritten for each time period independently as the succession of two linear layers. If an L2 reconstruction loss is used between the output and the input, these two successive linear layers are equivalent, up to a simple transformation of the feature space, to performing PCA decomposition followed by reconstruction [60], [61]. In this case, our architecture is equivalent to performing PCA on each time period independently, a property that we verified experimentally: for each time period $\mathbf{y}$, we reconstruct input feature $\mathbf{x}$ in the basis given by the PCA of image features in that period. In contrast to PCA, however, our bilinear module can naturally be inserted in an encoder-decoder architecture. This in turn allows training the entire factored image representation (including the bilinear module, encoder and decoder) in an end-to-end manner using more complex loss functions that go beyond simple L2 reconstruction.

4.2.2 Relation to bilinear style and content factorization

By representing vectors as a bilinear combination of time and latent content variables, our model is related to the bilinear style and content separation of [1]. Our approach has, however, three fundamental differences. First, we assume the style (in our case time) vector is known, given and fixed for each input instance, and thus we only need to estimate the content vector. Second, because we assume the style is known, we can estimate the content vector with a simple linear operation. This is in contrast to [1] who have to resort to an algorithm based on an iterative SVD. Because of this key difference the model of [1] also requires multiple observations of the same content with different styles, i.e. paired data, which our model does not and can operate on unpaired data collections. Finally, we include our bilinear separation approach as a module in a CNN architecture, which allows for end-to-end parameter learning.

4.2.3 Relation to concatenation-based generation methods

The standard way to combine two variables in a CNN architecture is to concatenate them (or their transformed versions. See for example [62]). Since the variables are often estimated independently in a feed forward way, we omit the index $\mathbf{y}$ for the $\mathbf{z}$ variable in this paragraph. The output of a decoder $g(\mathbf{z}, \mathbf{y})$ in the form of a single fully connected layer applied to a concatenation of input features z and y is written as

$$\mathbf{x}^k_{\mathbf{y}} = g(\mathbf{z}, \mathbf{y})^k = \sum_{i=1}^{S} w^{i,k}\mathbf{z}^i + \sum_{i=S+1}^{S+T} w^{i,k}\mathbf{y}^i, \tag{4}$$

where $\mathbf{w}$ are the weights of the fully connected layer. This equation is to be compared with eq. (2), where the interaction between $\mathbf{y}$ and $\mathbf{z}$ is multiplicative and not additive.

While the output is in practice often a result of several layers, it builds on this simple additive combination of the latent style and time. The resulting architecture is visualized in figure 3b, and we compare its representational power to our bilinear module in section 6.

5 Architecture and loss for image translation

We propose two different network architectures. The first one, described in section 5.1, is a standard auto-encoder architecture that uses our bilinear module as its bottleneck. We use it to show that our bilinear module works better than a baseline concatenation. The second one, described in section 5.2, includes our bilinear factorization module in a modern image translation model. The architecture is based on CycleGAN [3], but leveraging our bilinear module learns a single encoder/decoder for all time periods instead of having a specific encoder/decoder for each pair of source and target periods.

5.1 Bottleneck auto-encoder

Architecture. Standard auto-encoder architectures first reduce the dimensionality of the input image to a low dimensional feature vector using a succession of convolutions with non-linearities, subsampling and max-pooling then increase the dimensionality again using up-convolutions. In our experiments we used an architecture similar to [22] for the *conv4* features of an AlexNet [63], which we pre-trained on ImageNet [64]. In the middle of the network, we introduce a factorization module as described in section 4 and visualized in figure 3. In the following discussion of the training loss, we denote our complete bottleneck auto-encoder architecture as generator M.

Training loss. We performed experiments by training with (i) only a reconstruction loss or (ii) a combination of reconstruction and adversarial losses. We found the latter leads to visually more pleasing results and outline the corresponding training procedure next. More precisely, we train a fully convolutional network D, as a discriminator, that takes as input either a real image I from the dataset, or an image generated by our model $M(I, y_I)$, where y_I is the time period corresponding to image I, and evaluates for each location of the image, in a convolutional way, if it comes from the dataset or our model. The generator M is trained to both provide a good reconstruction and confuse the discriminator. Our full loss for fine-tuning the network is a combination of the L_1 reconstruction loss and the adversarial loss:

$$\mathcal{L}_{total}(M) = \mathop{\mathbb{E}}_{I}\left[\|M(I, y_I) - I\|_1\right] + \lambda\mathcal{L}_{adv}(M), \tag{5}$$

where M is our bottleneck auto-encoder model, D the discriminator, the expectation is taken over the training set of images I with associated time stamps y_I, and λ is a hyper-parameter balancing the total loss between reconstruction accuracy and image realism.

To improve the stability of the training, we use the least-squares generative adversarial network loss, as described in [65]. The adversarial loss for the generator thus is:

$$\mathcal{L}_{adv}(M) = \mathop{\mathbb{E}}_{I}\left[(D(M(I, y_I)) - 1)^2\right], \tag{6}$$

where $D(M(I, y_I))$ is the output of the discriminator for the generated image $M(I, y_I)$. In other words, the generator tries to produce images that are indistinguishable from real images by the discriminator, i.e. the discriminator outputs label 1. Note that this adversarial loss also depends on the discriminator D, which is trained jointly with our model using the following loss function

$$\mathcal{L}_{disc}(D) = \mathop{\mathbb{E}}_{I}\left[D\left(M(I, y_I)\right)^2 + (D(I) - 1)^2\right], \tag{7}$$

where the expectation is taken over the set of training images I with associated time stamps y_I. The first term, $D\left(M(I, y_I) - 0\right)^2$, is the loss on generated images, which the discriminator is trying to map to class 0, and the second term, $\left(D(I) - 1\right)^2$, is the loss on real images I, which the discriminator is trying to map to class 1. The intuition is that the discriminator tries to separate real and generated images while the generator tries to produce images that are indistinguishable from real photographs. Note that losses given by (6) and (7) are applied in a fully convolutional manner as described in [3].

Discussion. The size of the bottleneck inside the generator M, i.e. code $\mathbf{z} = g(\mathbf{x}, \mathbf{y})$ in the bilinear module, is a critical parameter for such an approach. It influences the result in two different ways. Decreasing the dimension of $\mathbf{z}$ corresponds to decreasing the dimension of the linear space on which $\mathbf{x}$ is projected before being reconstructed into $\mathbf{x_y}$. On one hand, as a result of the information loss corresponding to this projection, the reconstruction error $\|\mathbf{x} - \mathbf{x_y}\|$ will be larger, and the reconstructed image will lack details. On the other hand, when the size of the object specific code $\mathbf{z}$ is reduced, the network has to rely more on the time specific code $\mathbf{y}$ to reconstruct the training data. As a result, the reconstructed images will exhibit more strongly the characteristic features of the images in the target period, which is part of our goal. In the extreme case where only the time specific code $\mathbf{y}$ is preserved and the dimension of the bottleneck $\mathbf{z}$ is zero, the best the network could do in terms of the reconstruction loss on the training data is to produce an output close to an average training image for the given time period. We find that for simple datasets, such as for faces, a balance between preserving image quality and generating typical images for each period can be found. However, in other cases, such as with our car dataset, we found that with this architecture it is not possible to produce strong appearance variations when changing the time period while preserving the characteristic features of the particular object depicted in the image.

5.2 Adversarial translation

Architecture. A successful approach to image translation that does not use a standard auto-encoder architecture with a bottleneck, CycleGAN, was recently introduced in [3]. The work is focused on the case of two image domains, such as two different time periods in our case. The network, based on ResNet [66], includes shortcut connections that prevent information from being lost and preserve image details.

One of the main drawback of this approach, is that it does not build a shared factored representation, and thus requires to train two translation networks per domain pair. It also does not allow to represent continuous quantities, such as time. We extend the architecture of [3] to multiple domains and continuous variables. To achieve this we (i) share the network weights for all translations, and (ii) add a factorization module as described in section 4 as a central block, in parallel to a shortcut connection.

The dimensionality reduction is not necessary in the cycleGAN architecture because of the domain adversarial loss. We can thus remove the bottleneck in our bilinear factor separation, which makes it equivalent to a single bilinear layer. Interestingly, while the architecture with a single bilinear layer is simpler than the full bilinear module, it typically has more parameters since it does not have a bottleneck.

Training Loss. To train our translation architecture denoted M, we use a combination of multiple losses that we outline next. First, like with the auto-encoder, we use a reconstruction loss:

$$\mathcal{L}_{id}(M) = \mathop{\mathbb{E}}_{I}\left[\|M(I, y_I) - I\|_1\right], \tag{8}$$

where the expectation is taken over all images I and y_I is the time period associated to image I. Similar to [3], we also consider a cycle consistency loss

$$\mathcal{L}_{cycle}(M) = \mathop{\mathbb{E}}_{I,y}\left[\|M(M(I, y), y_I) - I\|_1\right], \tag{9}$$

where the expectation is taken over all images I and all possible target time periods y, and y_I is the time period associated to image I. This loss ensures that a source image I translated to time y, $M(I, y)$, can be converted back to the source image I when using the source time y_I as the input to the generator.

Finally, to enforce that the generated images look different for each time period, we rely on an adversarial loss

$$\mathcal{L}_{adv}(M) = \mathop{\mathbb{E}}_{I,y}\left[(1 - D^y(M(I, y)))^2\right], \tag{10}$$

where the expectation is taken over all images I and all possible target time periods y. The intuition is that the generated image $M(I, y)$ for time period y should be classified correctly by the discriminator D. We train a single discriminator D that takes an image as input and is trained to output a vector that contains one hot encoding of the input image time period. D^y denotes the component of the output vector corresponding to time y. The discriminator is trained jointly with the generator M using the following loss

$$\mathcal{L}_{disc}(D) = \mathop{\mathbb{E}}_{I}\left[(1 - D^{y_I}(I))^2\right] + \lambda_{disc} \mathop{\mathbb{E}}_{I,y}\left[D^y(M(I, y))^2\right], \tag{11}$$

where the first term encourages that the discriminator predicts 1 for the correct time period y_I for training images I. The second term encourages that for the generated images $M(I, y)$ the discriminator output for period y, D^y should be zero. In other words, the discriminator should be able to detect that the generated image is "fake", i.e. should not belong to the target period.

The final training loss is the weighted sum of reconstruction, cycle consistency and adversarial losses:

$$\mathcal{L}_{total}(M) = \mathcal{L}_{id}(M) + \lambda_{cycle}\mathcal{L}_{cycle}(M) + \lambda_{adv}\mathcal{L}_{adv}(M), \tag{12}$$

where the λs are the relative weights of each loss. Note that in CycleGAN, only two classes are considered, while we can consider an arbitrary number of classes (in our case time periods) using a single network.

5.3 Implementation details

We implement our module and networks in Torch [67]. We modified the implementation of the bilinear layer by reordering the loops to reduce the number of (expensive)

	cars (3 periods)			faces (3 periods)		
	Same year	Modif. year	L1 error	Same year	Modif. year	L1 error
	Classification		Reconstruction	Classification		Reconstruction
Real images	93%	*n.a.*	*n.a.*	96%	*n.a.*	*n.a.*
Bilinear auto-encoder (64)	84%	69%	0.13	95%	15%	0.08
Bilinear auto-encoder (16)	86%	80%	0.16	99%	90%	0.11
Concat. auto-encoder (64)	55%	23%	0.13	80%	11%	0.10
Concat. auto-encoder (16)	64%	48%	0.16	81%	9%	0.11
Concat. auto-encoder (4)	64%	64%	0.20	96%	81%	0.14
Bilinear translation (ours)	83%	63%	0.017	97%	91%	0.008
Concat. translation	83%	63%	0.016	98%	88%	0.009
CycleGAN [3]	76%	70%	0.015	97%	87%	0.009
StarGAN [59]	92%	40%	0.044	99%	52%	0.030

TABLE 1
Quantitative evaluation of our models and baselines for both the cars [4] and faces [5] datasets. We evaluate both a reconstruction error (normalized L1 loss, lower is better) and a "inception" type of score (percentage of good classification, higher numbers are better)

calls to the BLAS routines to the size of the smallest dimension. We will share this code upon publication. With this implementation, each iteration of our bilinear layer is as fast as the baseline concatenation module. For our auto-encoder network, it takes 3 minutes to do a full pass on a dataset of 10,000 images on a GTX 1080 GPU, i.e. a full pass on 1M images would take about 5 hours. Furthermore, compared to a concatenation baseline with a similar number of parameters, our model converges in a smaller number of iterations to a lower value of the objective.

6 Results

In this section, we begin by presenting the datasets we use (section 6.1). Then, we provide quantitative and qualitative results corresponding to our three main contributions. First, in section 6.2 we analyse the effect of different architectures and losses on the results of image translation. Second, in section 6.3 we provide an in depth comparison of the standard concatenation and our new bilinear factorization module. Finally, in section 6.4 we give examples of the new types of image analysis enabled by our model: generating history-lapse videos and discovering trends by analyzing changes across time.

6.1 Datasets and metrics

To illustrate the generality of our method, we use two datasets of historical images for our evaluation. The historical car dataset [4] is a set of cars made in a span of 80 years. The historical yearbook portraits [5] contains pictures of students taken across 80 years. For both datasets, we grouped the images by decades and selected three decades with clear differences to analyse the results of our method (40s, 70s and 2000s for the faces and 40s, 60s and 90s for the cars). We thus obtained 3600 training images and 300 validation images for the faces dataset across three decades, and 7299 training images and 1258 validation images for the cars dataset. We also introduce appropriate metrics to quantitatively compare the results of the different approaches.

Historical car dataset. We focus the paper on the challenging car dataset introduced in [4], containing modern images of cars together with their construction date, between 1920 and 1999. This dataset presents several difficulties, with cars having very different appearances and backgrounds. The cars are also imaged in a variety of lighting conditions and from a variety of viewpoints. To localize the cars in the image, we first run a standard Faster R-CNN car detector [68] pre-trained on Pascal VOC images. The detected bounding boxes are then resized to 227x227 pixels. Despite this pre-processing, this dataset remains a great challenge.

Historical yearbook portraits. We also applied the proposed model to the historical photograph collection of American high school yearbook portraits from 1930s to 2000s of [5]. Aligned face images are known to be relatively easy to analyze, and results on this type of data have been demonstrated in, e.g., [5], [51], [52]. We slightly change our loss function for this dataset, in the case of the bottleneck auto-encoders. While fine-tuning our model, we apply a mask that increases the weight of both the reconstruction loss and the adversarial loss around the eyes of the faces to have sharper results.

Evaluation metrics. Quantitative evaluation for our task is difficult as there is no ground truth for how a specific car or portrait looked like in both 1920s and 1990s. We evaluate the different methods by looking at a reconstruction metric and a classification metric. Our reconstruction metric is simply the L1 distance between the input image and its translation into its ground truth period. While this is not a very precise metric of image similarity, we found that for our task it correlated well with perceived image quality. As pointed above, this reconstruction metric is limited since it can only be used together with the ground-truth time period, and doesn't evaluate whether the generated image shows characteristics of the target time period. This is the goal of our classification metric. Following [58], we use an off-the-shelf classifier to assess whether our model is able to generate images that "look" like the images of the intended time period. This metric is related to the "inception score" [69], object detection evaluation in [70] and the "semantic interpretability" measure in [71]. In detail, we trained a classifier on the *conv4* features of an AlexNet network trained on ImageNet [64] to predict time periods on real images and applied it (i) to a validation set of previously unseen real images, and (ii) to images generated by the different methods.

For each model and dataset, we report two numbers:

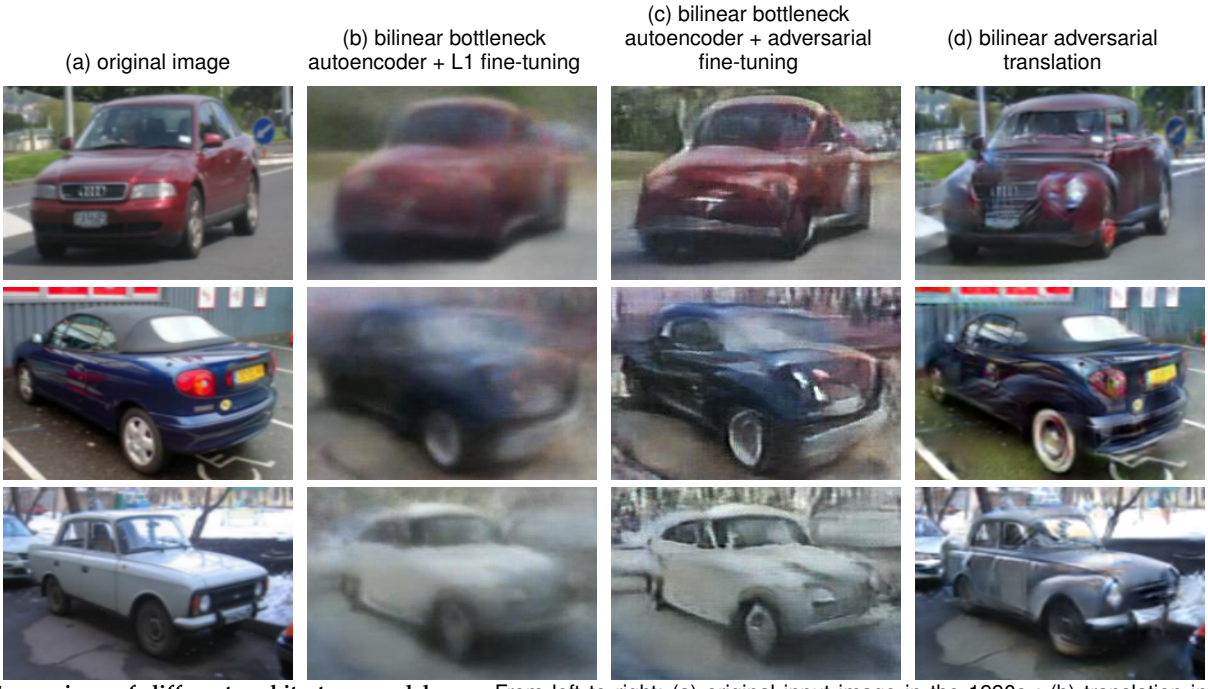


Fig. 4. **Comparison of different architectures and losses.** From left to right: (a) original input image in the 1990s ; (b) translation into 1940s generated with a bilinear bottleneck autoencoder fine-tuned using only a L1 reconstruction loss produces rather blurry output ; (c) Fine-tuning using an additional adversarial loss produces sharper results ; (d) the best results are obtained using the bilinear adversarial translation model.

(i) the percentage of correctly classified images in the case when the input image is translated into its ground truth time period ("Same year"), and (ii) the percentage of correctly classified images when the input image is translated to a different time period ("Modified year"). The first score evaluates whether the model preserves the temporal characteristic of the image when the time period is unchanged. The second score evaluates whether the model manages to change the time characteristics of the generated image in a way that fools the classifier. Higher numbers are better for both metrics; they can be interpreted as a success of the translation model to change the time period of the input image. All quantitative results are are summarized in table 1 and discussed in sections 6.2 and 6.3.

6.2 Analysis of architectures and losses for image translation

In this section we analyze the different architectures and losses for the task of temporal image translation. We begin with the analysis of visual quality of the results, then perform quantitative analysis, and finally discuss the typical failure modes.

Qualitative analysis. We focus our qualitative analysis on the more challenging car dataset. Qualitative results on the face dataset are shown in section 6.4 and on the project webpage [72]. First, in figure 4, we compare the results of translation of cars from the 1990s to the 1940s using our bilinear factorization module inserted into the bottleneck auto-encoder architecture without (b) and with (c) adversarial loss as well as inserted into the adversarial translation framework (d). The results in column (b) show that using only the L1 reconstruction loss with a bottleneck auto-encoder architecture results in blurry images that show some changes related to the time period but are hard to interpret. As shown in column (c), adding an adversarial loss to the bottleneck auto-encoder produces more realistic images with clearer differences between the time periods. However, there is still significant loss in image quality, due to the dimensionality reduction in the bottleneck. Finally, images produced with the adversarial translation (column (d)) are the most realistic and make changes over time easiest to interpret.

Second, in figures 5 and 6, we show a comparison between our bilinear adversarial translation model, and StarGAN [59] and CycleGAN [3], which are the state-of-the-art models for image translation. For CycleGAN, we have trained three pairwise models, in order to cover transformations from and to each time period. Note that our bilinear adversarial model is only trained once with the three time periods. While the images are slightly different, the overall image quality and the temporal changes are similar. However, our bilinear adversarial translation architecture has the advantage of being a single model that is faster to train, as most of the weights of the model are shared. Note that the adversarial translation using a concatenation based factorization module produces similar quality of results and is not shown in the figure. Qualitative results in figure 5 and 6 also include results of the models with unmodified years. As expected the different methods make little change to the original images.

In order to provide a comparison between our results and StarGAN [59], we ran the StarGAN algorithm on the face and car datasets that we use in our submission. We use the default parameters of the StarGAN implementation provided by [59]. We use the same settings as used in [59] for the RaFD dataset, which similar to our datasets features a single attribute with multiple possible values. It is

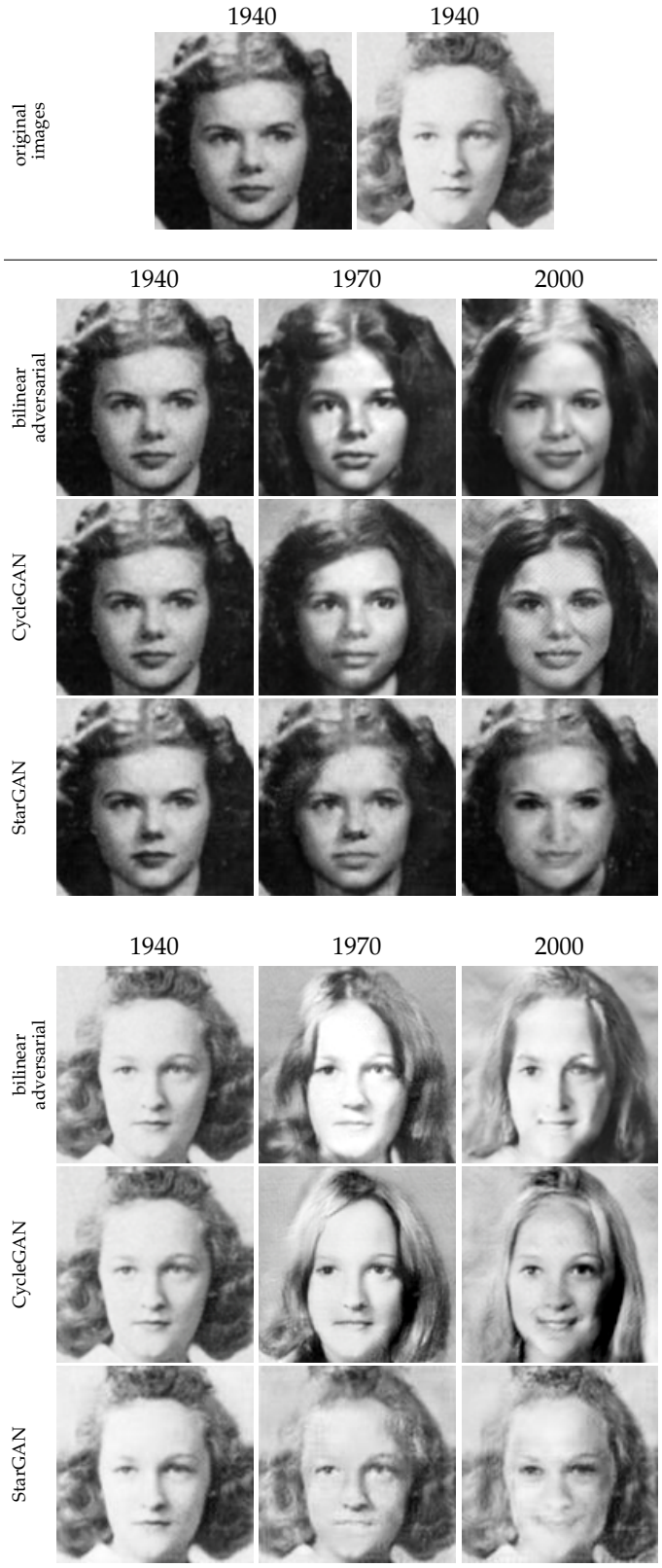


Fig. 5. **Comparison between our bilinear adversarial translation model, CycleGAN [3] and StarGAN [59].** Original input images (all from 1940s) are shown in the top row. Their translations to different time periods, including their original time period, are shown below. Our bilinear adversarial translation and CycleGAN produce sharp images and clear changes over time with comparable visual quality of results. However, our bilinear adversarial translation architecture has the advantage of being a single model whereas a separate CycleGAN model needs to be trained for each target time period. StarGAN learns to change facial features but not the hairstyle.

Fig. 6. **Comparison between our bilinear adversarial translation model, CycleGAN [3] and StarGAN [59].** Original input images (all from 1960s) are shown in the top row. Our bilinear adversarial translation and CycleGAN produce sharp images and clear changes over time with comparable visual quality of results. StarGAN did not learn to produce significant changes over time on the car dataset.

interesting to observe that on the face dataset (fig. 5) the StarGAN results are qualitatively different to our results and results of CycleGAN. StarGAN mainly changes facial expressions of the faces, progressively adding more smile over the years, but does not significantly change the other attributes such as the hairstyle, which are also modified by our approach (bilinear adversarial) and CycleGAN. For the more challenging car dataset (fig. 6), we found that StarGAN did not learn to produce significant changes to the images over time. We think that this could be attributed to the increased difficulty of the dataset that includes non-aligned images of cars depicted from significantly different viewpoints and under different illuminations. This is in contrast to the face dataset, where images are fairly well aligned and captured from a similar viewpoint and under similar imaging conditions.

We note that some of our resulting images still show artifacts. While local changes in textures look often convincing, the artifacts are mostly present when the model tries to change the overall shape of the object. Similar artifacts are present in the output of the strong CycleGAN baseline we compare with. This further underscores the difficulty of the image translation task on complex non-aligned images with significant changes in appearance.

Quantitative analysis. The visual results are supported by the quantitative results reported in table 1. First, for both datasets, there is a clear gap between reconstruction errors for methods using adversarial translation and the bottleneck auto-encoder, with more than an order of magnitude difference. This confirms the visual results form figure 4. The reconstruction error for the bottleneck auto-encoder architecture is improving with the size of the bottleneck, but does not reach the quality of the result of the adversarial translation model. Second, the classification results, which are much higher than chance for all adversarial translation frameworks and for some bottleneck auto-encoder architectures demonstrate that both the auto-encoder and adversarial translation models can effectively change the appearance of the input image towards the target period. Third, we note that StarGAN did not perform as well as the other methods on the classification metric, as it seems to learn only limited amount of appearance changes as discussed in more detail in the qualitative results above. Finally, the joint results of the three independently trained pairwise CycleGAN models are similar to the results of our single bilinear adversarial translation model, which confirms the visual results from figure 6. The adversarial translation models, and in particular our bilinear adversarial translation model, show both a high classification score and a low reconstruction error. This makes these models good candidates to perform visual discovery in unpaired temporal image collections, as we will demonstrate in section 6.4.

Failure cases. Figure 7 shows several typical failure examples of our best performing approach, the bilinear adversarial translation model. In general, our method fails on uncommon images, where it typically produces a car that is nearly identical to the original input image, with sometimes localized changes on small parts such as the wheels. In other words the model defaults to the original input image when it doesn't know how to modify the image correctly. These failure cases typically correspond to unusual viewpoints (figure 7(a)), open trucks or boots (figure 7(b)), or rare or otherwise unique cars in our dataset (figure 7(c)).

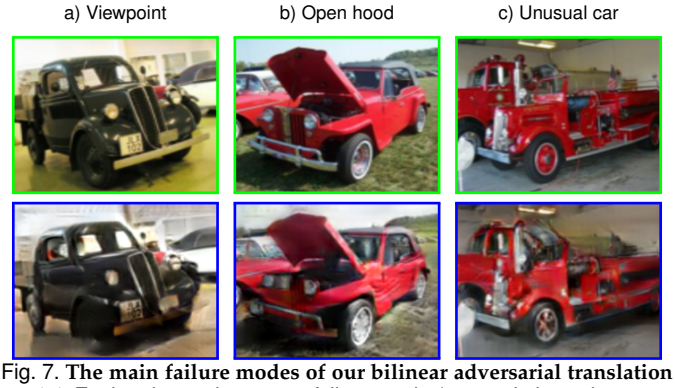


Fig. 7. **The main failure modes of our bilinear adversarial translation model.** Each column shows one failure mode (unusual viewpoint, open hood, rare car type). In each column the top image shows the input (car from the 1940s, green border) and the bottom image shows the output translation into the 1990s (blue border). In all cases the model fails to modify the input image in a substantial way, but often still modifies some small parts, such as the wheels or the headlights.

6.3 Comparison of bilinear and concatenation factorization modules

In this section, we analyze the differences between the standard concatenation based factorization module and our new bilinear factorization module. We focus this comparison only on the simple bottleneck auto-encoder architecture. In the more powerful framework of adversarial translation, both types of factorization modules produce similar results, which we believe is due to the higher capacity of the base network in the adversarial translation translation architecture that uses more layers and residual blocks bypassing the need for explicit modelling of multiplicative interactions.

Qualitative analysis. As in the previous section, we focus our qualitative analysis on the more challenging car dataset and provide results on the face dataset on the project webpage [72]. Qualitative translation results for different code sizes are shown in figure 8 and demonstrate clear differences between the concatenation-based and bilinear models. Indeed, for the code size 16 and 64, the image quality is similar for both types of factorization modules, but the concatenation based module produces almost no differences between the two target time periods, while very visible changes happen using the bilinear module. By reducing the code size of the concatenation based module further, for example to 4, it is possible to force the model to produce some changes, but only at the cost of very low quality images, where a large part of the image identity has been lost. These results clearly demonstrate the advantage of the bilinear factorization module that we introduced and analyzed in section 4.

Quantitative analysis. The visual analysis is supported by the the quantitative results reported in table 1. The

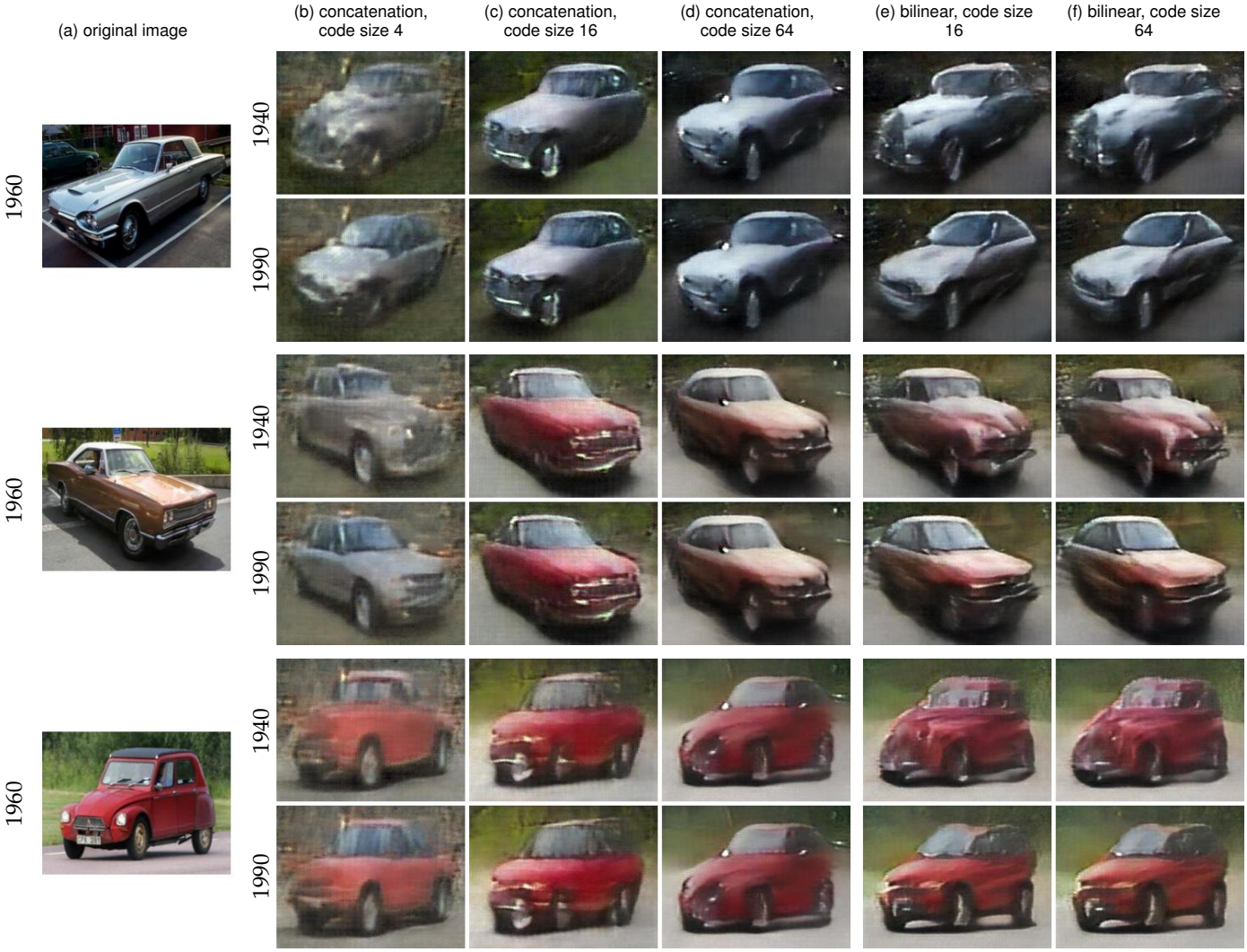


Fig. 8. **Comparison of bilinear and concatenation modules in the bottleneck auto-encoder architecture.** In each example we show: (a) input original image (1960s); (b-d) concatenation-based translation to 1940s (top) and 1990s (bottom) using different code size of 4 (b), 16 (c) and 64 (d); (e-f) bilinear translation to 1940s (top) and 1990s (bottom) using code size of 16 (e) and 64 (f). Note that concatenation modules produce either limited temporal changes (larger code size) or low quality reconstruction (small code size). On the contrary, the bilinear factorization module (e-f) produces significant temporal variations with reasonable image quality for a range of code sizes.

reconstruction error is similar using both factorization modules with the same code size, but the classification scores are much higher, on both datasets, using our bilinear module. Obtaining higher classification scores using the concatenation-based model is possible only at the cost of strongly decreasing the code size, and thus reducing the image quality. However, even with a code size of 4, resulting in an important decrease of the reconstruction error, the classification scores of the concatenation module are still lower than the scores of our bilinear module. This demonstrates the clear advantage of our bilinear factorization module when employed in the bottleneck auto-encoder architecture. This is because the auto-encoder has to both (i) preserve the quality of the original image (which requires large size of the bottleneck) and (ii) change the appearance of the image to match the target time period (which is enforced by making the bottleneck small so that the decoder has to make use of the input time variable y). It is hard to achieve both these conflicting objectives (i) and (ii) with the concatenation architecture. In contrast, the bilinear module addresses this issue as it explicitly encourages the separation of the appearance specific to time and the appearance that is independent of the time. As a result, including the bilinear module in the auto-encoder architecture works much better as it allows to use larger sizes of the bottleneck (and hence better reconstruction quality) and yet forces the model to modify the appearance of the input image when changing the time period by modelling each time period with a separate subspace in the feature space as discussed in section 4.2.1.

6.4 Visual discovery via bilinear image translation

In this section we demonstrate how to apply our bilinear adversarial translation model for the task of visual discovery in an unpaired image collection annotated with time

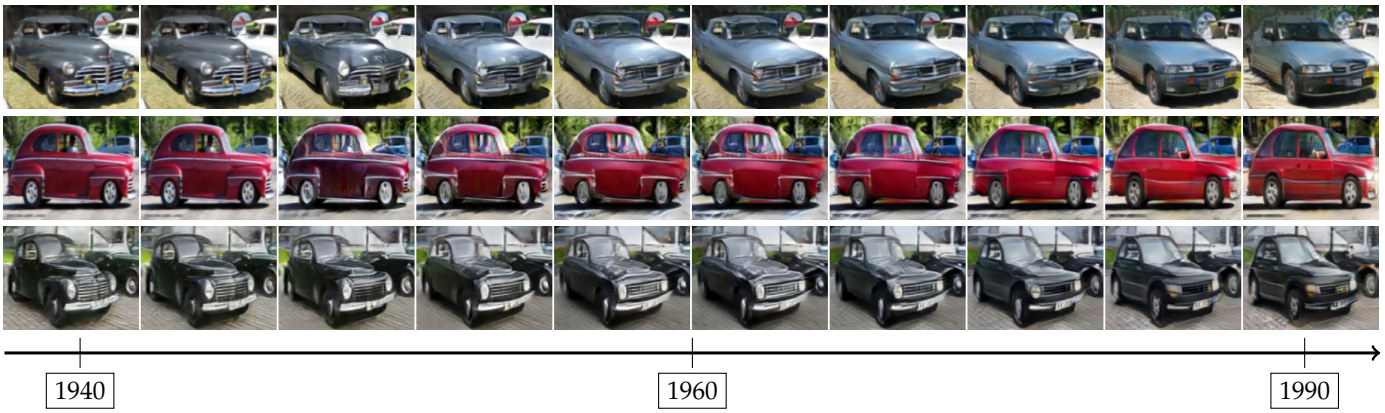


Fig. 9. Examples of frames from history-lapse videos generated by our bilinear adversarial translation model. Here, input images of cars from the 1940s (left) are translated to different future times (left to right) by varying the input time variable. The output is a video showing changes of car style over time. Please note how the shape and appearance of the input cars changes over time. For example, the curved hood gradually fattens, the round lights become rectangular, the windscreen gets bigger, the front grille gets smaller and the running board on the side gradually disappears. Note also that other characteristics of the input car instances, such as viewpoint, color and the overall shape are preserved by the learnt temporal transformation. History-lapse videos can be found on the project webpage [72]

stamps. First, we train a bilinear translation model on the input image collection using the time stamp of each image as its (observed) time variable. Note that the model is able to extract temporal changes over time from the unpaired data, i.e. despite never seeing a particular object instance (e.g. a car or a face) evolve over time. Then, we apply the trained model on a new unseen test image. By varying the input time stamp we generate a history-lapse video that translates the input image into different time periods while preserving the identify of the depicted object. Finally, we analyze the generated image sequence to extract interesting changes in style over time.

6.4.1 Creating history-lapse videos

Given a pre-trained bilinear adversarial translation model we generate a history-lapse for a new test image. This is achieved by varying the input time variable. In detail, using soft-assignment of the year y to define the time vector $\mathbf{y}$, we produce history-lapse videos with dense sampling of time. We generate videos at a sampling rate of 1 frame per year. **Please see the videos on the project webpage [72].** Example frames are shown in figure 9. To generate these figures, we use a soft assignment of the time t into an embedding $\mathbf{y}$. It is a continuous mapping of the time t, defined by:

$$y_i(t) = K_t \exp\left(\frac{(i-\alpha t)^2}{2\sigma^2}\right), \tag{13}$$

where σ and α are hyper-parameters and K_t is a normalization factor set so that $\|\mathbf{y}(t)\|_2 = 1$. This mapping provides a soft discretization of continous time into bins representing different time periods. We use concatenation to combine the code $\mathbf{z}$ and the year embedding $\mathbf{y}$. The videos reveal interesting changes of car style learnt from the entire dataset but applied on a specific instance of a car depicted from a given fixed viewpoint and captured in given imaging conditions. Note how some parts appear, disappear or are transformed over time. For example, the running board on the side of the car, characteristic for the 1940s gradually disappears over time; the round lights of the 1940s are transformed into rectangular ones in the 1990s; or the windows, initially small, are enlarged over time. Note that the model learns to generate such changes without annotated part correspondence in the training data or seeing a specific object instance evolve over time in training. Please note also that such translation with a continuously changing attribute (here the time variable) would not be possible using the standard CycleGAN approach [3].

6.4.2 Analyzing trends over time

An analysis of appearing, evolving and disappearing parts can be performed by looking at the differences between the images generated at the different time periods and the original input image. This way we can highlight the most important differences for a specific generated time period. Figure 10 shows such difference images for several examples from the car dataset. Please note the consistent changes over time. For example, the strong image differences (bright yellow color) at the base of windows and windshields indicate the evolution of their size over time, clearly showing the trend of having larger windows and windshields in later time periods. This trend is consistently exhibited in the input dataset on many different cars. Some elements are also characteristic for a specific period. For example, the running board on the side of the car is often present in the 1940s, but disappears (and hence is highlighted in bright yellow in the difference images) in the later periods. The wheels and headlights are also elements that have characteristic appearance at a specific time period and are often highlighted in the difference images.

Similarly, figure 11 reveals the trends in hairstyle and facial expression across time in women's yearbook portraits.

Please note how the vocabulary of parts (wheels, headlights or windshields for cars and hair, eyes or mouth for faces) as well as their correspondence across different instances is learnt implicitly by our model from the unpaired training data. The outcome is that the model is able to synthesize appearance variation over time for a specific new input object instance despite changes in viewpoint or specific new appearance of the input car. Note also that our model enables *pixel-level* analysis of temporal trends by just comparing the synthesized time-lapse image sequences.

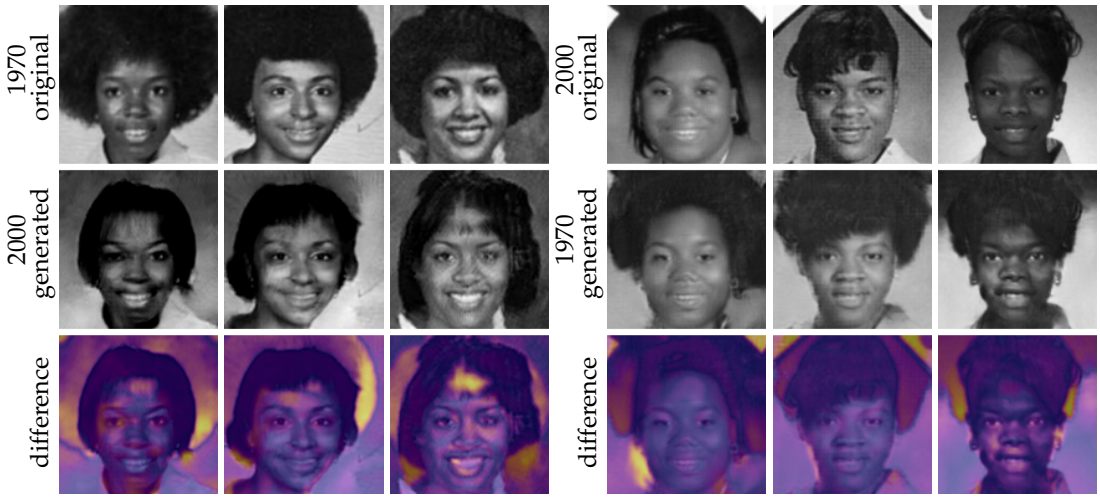
(a) Input images of cars from the 1960s (top row) and their generated translations into 1990s (middle row). The difference images (bottom row) highlight the changes in the headlights of the cars, going from small round ones to large rectangular ones, as well as the increasing size of the car windows and windshield.

(b) Input images of cars from the 1940s (top row) and their generated translations into 1990s (middle row). In addition to the changes in the shape of headlights and the size of windows, similar to (a), the difference images (bottom row) highlight the disappearing metallic bumpers as well as the evolution of the shape of the hood, which flattens over time.

(c) Input images of cars from the 1990s (top row) and their generated translations into 1940s (middle row). The difference images (bottom row) highlight the appearing running board (or footboard) as well as the changing appearance of wheels.

Fig. 10. **Analyzing trends over time.** Each plate (a-c) shows consistent changes over time. In each plate, we show the original input images (top row), their translation into another period (middle row) and the absolute differences (bottom row) between the top and middle images. The absolute differences are shown as heatmaps where bright yellow color indicates the maximum absolute difference within the image, while dark blue colors correspond to small differences. We superimpose the heatmaps on a low contrast version of the original input image to further highlight the changes produced by our model. Please see more results on the project webpage project webpage [72]

(a) Input portraits from the 1940s (top row) and their generated translations into 2000s (middle row). The difference images (bottom row) reveal consistent changes in haircuts, with hair getting longer, and less curly in the 2000s.



(b) Input portraits from the 1970s (top row) and their generated translations into 2000s (middle row). The difference images (bottom row) reveal a particular trend in the hairstyle of African American women, who often had a voluminous afro haircut in the 1970s, which mostly disappeared in the 2000s.

(c) Input portraits from the 1940s (top row) and their generated translations into 2000s (middle row). In addition to changes in haircut, the difference images (bottom row) reveal changes in facial expression with bigger smiles showing more teeth in the 2000s as well as the color of the lips getting lighter, possibly because of changes in the popular lipstick color and use.

Fig. 11. **Analyzing trends over time.** Each plate (a-c) shows consistent changes over time. In each plate, we show the original input images (top row), their translation into another period (middle row) and the absolute differences (bottom row) between the top and middle images. The absolute differences are shown as heatmaps where bright yellow color indicates the maximum absolute difference within the image, while dark blue colors correspond to small differences. We superimpose the heatmaps on a low contrast version of the original input image to further highlight the changes produced by our model. Please note consistent changes in hairstyle in plates (a) and (b), and smile in plate (c).

7 Conclusion

We have presented and demonstrated the concept of temporal visual discovery via bilinear image translation. We have introduced a new bilinear module for learning factored image representations, and shown that it is superior to existing concatenation-based architecture currently used for factored image synthesis using bottleneck auto-encoders. We have also shown how the new module can be included in an adversarial translation architecture for efficiently handling several target time periods. Finally, we have demonstrated that our model can identify trends over time in a challenging dataset depicting non-aligned car images.

Acknowledgement

This work has been supported by the European Research Council (ERC grant LEAP no. 336845), Agence Nationale de la Recherche (Semapolis project, ANR-13-CORD-0003), ANR project EnHerit ANR-17-CE23-0008, CIFAR Learning in Machines&Brains program and European Regional Development Fund under the project IMPACT (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000468).

References

[1] Tenenbaum, J.B., Freeman, W.T.: Separating style and content with bilinear models. Neural computation **12**(6) (2000) 1247–1283
[2] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR. (2015)
[3] Zhu, J.Y., Park, T., Isola, P., Efros, A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proc. ICCV. (2017)
[4] Lee, Y., Efros, A., Hebert, M.: Style-aware mid-level representation for discovering visual connections in space and time. In: Proc. ICCV. (2013)
[5] Ginosar, S., Rakelly, K., Sachs, S., Yin, B., Efros, A.: A century of portraits: A visual historical record of american high school yearbooks. arXiv preprint arXiv:1511.02575 (2015)
[6] Michel, J.B., Shen, Y., Aiden, A., Veres, A., Gray, M., et al.: Quantitative analysis of culture using millions of digitized books. Science **331**(6014) (2011)
[7] Roy, D., et al.: The human speechome project. Symbol Grounding and Beyond (2006) 192–196
[8] Marcus, D., Fotenos, A., Csernansky, J., J., M., Buckner, R.: Open access series of imaging studies (oasis): Longitudinal mri data in nondemented and demented older adults. Journal of Cognitive Neuroscience **22**(12) (2010) 2677–2684
[9] Tang, A., et al.: Canadian association of radiologists white paper on artificial intelligence in radiology. Canadian Association of Radiologists Journal **69**(2) (2018) 120 – 135
[10] Crowley, E.J., Zisserman, A.: Of gods and goats: Weakly supervised learning of figurative art. In: Proc. BMVC. (2013)
[11] Shen, X., Pastrolin, I., Bounou, O., Gidaris, S., Smith, M., Poncet, O., Aubry, M.: Large-scale historical watermark recognition: dataset and a new consistency-based approach. arXiv preprint arXiv:1908.10254 (2019)
[12] : the visual arts data service. https://www.vads.ac.uk/index.php
[13] Doersch, C., Singh, S., Gupta, A., Sivic, J., Efros, A.A.: What makes paris look like paris? In: Proc. ACM SIGGRAPH. (2012)
[14] Doersch, C., Gupta, A., Efros, A.: Mid-level visual element discovery as discriminative mode seeking. In: NIPS. (2013)
[15] Lee, S., Maisonneuve, N., Crandall, D., Efros, A., Sivic, J.: Linking past to present: Discovering style in two centuries of architecture. In: International Conference on Computational Photography. (2015)
[16] Dalal, N., Triggs, B.: Histogram of Oriented Gradients for Human Detection. In: Proc. CVPR. Volume 2. (2005) 886–893
[17] Kirby, M., Sirovich, L.: Application of the karhunen-loeve procedure for the characterization of human faces. IEEE PAMI **12**(1) (1990) 103–108
[18] Turk, M.A., Pentland, A.P.: Face recognition using eigenfaces. In: Proc. CVPR, IEEE (1991) 586–591
[19] Vasilescu, M.A.O., Terzopoulos, D.: Multilinear analysis of image ensembles: Tensorfaces. In: eccv02, Springer (2002) 447–460
[20] Tang, Y., Salakhutdinov, R., Hinton, G.E.: Tensor analyzers. In: ICML (3). (2013) 163–171
[21] Denton, E., Chintala, S., Fergus, R., et al.: Deep generative image models using a laplacian pyramid of adversarial networks. In: NIPS. (2015)
[22] Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks. CoRR **abs/1602.02644** (2016)
[23] Cheung, B., Livezey, J., Bansal, A., Olshausen, B.: Discovering hidden factors of variation in deep networks. arXiv preprint arXiv:1412.6583 (2014)
[24] Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: ICLR. (2016)
[25] Quack, T., Leibe, B., Van Gool, L.: World-scale mining of objects and events from community photo collections. In: Proc. CIVR. (2008)
[26] Martin-Brualla, R., Gallup, D., Seitz, S.: Time-lapse mining from internet photos. ACM Trans. Graph. **34**(4) (2015)
[27] Matzen, K., Snavely, N.: Scene chronology. In: Proc. ECCV. (2014)
[28] Sivic, J., Zisserman, A.: Video data mining using configurations of viewpoint invariant regions. In: Proc. CVPR. (2004)
[29] Grauman, K., Darrell, T.: Unsupervised learning of categories from sets of partially matching image features. In: Proc. CVPR. (2006)
[30] Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th Annual International Conference on Machine Learning, ACM (2009) 609–616
[31] Lee, Y.J., Grauman, K.: Learning the easy things first: Self-paced visual category discovery. In: Proc. CVPR. (2011)
[32] Singh, S., Gupta, A., Efros, A.A.: Unsupervised discovery of mid-level discriminative patches. In: Proc. ECCV. (2012)
[33] Sivic, J., Russell, B.C., Efros, A.A., Zisserman, A., Freeman, W.T.: Discovering objects and their location in images. In: Proc. ICCV. (2005)
[34] Karlinsky, L., Dinerstein, M., Ullman, S.: Unsupervised feature optimization (ufo): Simultaneous selection of multiple features with their detection parameters. In: Proc. CVPR. (2009)
[35] Russell, B.C., Efros, A.A., Sivic, J., Freeman, W.T., Zisserman, A.: Using multiple segmentations to discover objects and their extent in image collections. In: Proc. CVPR. (2006)
[36] Todorovic, S., Ahuja, N.: Extracting subimages of an unknown category from a set of images. In: Proc. CVPR. (2006)
[37] Matzen, K., Snavely, N.: Bubblenet: Foveated imaging for visual discovery. In: Proc. ICCV. (2015)
[38] Juneja, M., Vedaldi, A., Jawahar, C.V., Zisserman, A.: Blocks that shout: Distinctive parts for scene classification. In: Proc. CVPR. (2013)
[39] Vittayakorn, S., Berg, A.C., Berg, T.L.: When was that made? In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE (2017) 715–724
[40] Palermo, F., Hays, J., Efros, A.A.: Dating historical color images. In Fitzgibbon, A., Lazebnik, S., Sato, Y., Schmid, C., eds.: ECCV (6). Volume 7577 of Lecture Notes in Computer Science., Springer (2012) 499–512
[41] Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnn models for fine-grained visual recognition. In: Proc. ICCV. (2015) 1449–1457
[42] Fukui, A., Park, D.H., Yang, D., Rohrbach, A., Darrell, T., Rohrbach, M.: Multimodal compact bilinear pooling for visual question answering and visual grounding. arXiv preprint arXiv:1606.01847 (2016)
[43] Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation **18**(7) (2006) 1527–1554
[44] Le Roux, N., Heess, N., Shotton, J., Winn, J.: Learning a generative model of images by factoring appearance and shape. Neural Comput. **23**(3) (Mar 2011) 593–650
[45] Osindero, S., Hinton., G.E.: Modeling image patches with a directed hierarchy of Markov random fields. In: NIPS. (2008)
[46] Salakhutdinov, R., Hinton, G.E.: Deep Boltzmann machines. In: AISTATS. (2008)
[47] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proc. ICML. (2008)

[48] Kulkarni, T., Kohli, P., Tenenbaum, J., Mansinghka, V.: Picture: A probabilistic programming language for scene perception. In: Proc. CVPR. (2015)
[49] Kulkarni, T., Whitney, W., Kohli, P., Tenenbaum, J.: Deep convolutional inverse graphics network. In: NIPS. (2015)
[50] Tatarchenko, M., Dosovitskiy, A., Brox, T.: Multi-view 3d models from single images with a convolutional network. In: Proc. ECCV, Springer (2016) 322–337
[51] Reed, S., Sohn, K., Zhang, Y., Lee, H.: Learning to disentangle factors of variation with manifold interaction. In: Proc. ICML. (2014)
[52] Yan, X., Yang, J., Sohn, K., Lee, H.: Attribute2image: Conditional image generation from visual attributes. Proc. ECCV (2016)
[53] Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. arXiv preprint arXiv:1603.08155 (2016)
[54] Wang, X., Gupta, A.: Generative image modeling using style and structure adversarial networks. arXiv preprint arXiv:1603.05631 (2016)
[55] Zhu, J.Y., Krahenbuhl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: Proc. ECCV. (2016)
[56] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS. (2015)
[57] Zhou, Y., Berg, T.L.: Learning temporal transformations from time-lapse videos. In: Proc. ECCV, Springer (2016) 262–277
[58] Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004 (2016)
[59] Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 8789–8797
[60] Baldi, P., Hornik, K.: Neural networks and principal component analysis: Learning from examples without local minima. Neural networks **2**(1) (1989) 53–58
[61] Sanger, T.D.: Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural networks **2**(6) (1989) 459–473
[62] Dosovitskiy, A., Springenberg, J., Tatarchenko, M., Brox, T.: Learning to generate chairs, tables and cars with convolutional networks. IEEE PAMI (2016)
[63] Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS. (2012) 1106–1114
[64] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, S., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., Li, F.: Imagenet large scale visual recognition challenge. IJCV (2015)
[65] Mao, X., Li, Q., Xie, H., Lau, R.Y.K., Wang, Z., Smolley, S.P.: Least squares generative adversarial networks. arXiv preprint arXiv:1611.04076 (2016)
[66] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
[67] Collobert, R., Kavukcuoglu, K., Farabet, C.: Torch7: A matlab-like environment for machine learning. In: BigLearn, NIPS Workshop. (2011)
[68] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. (2015) 91–99
[69] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: NIPS. (2016)
[70] Wang, X., Gupta, A.: Generative image modeling using style and structure adversarial networks. In: Proc. ECCV. (2016)
[71] Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: Proc. ECCV. (2016)
[72] : The project webpage. https://www.di.ens.fr/willow/research/bilineartranslation

Théophile Dalens received an MSc degree in computer science from Ecole polytechnique. He received a PhD degree from PSL university for his work at INRIA on image translation and visual discovery, under the supervision of Josef Sivic and Mathieu Aubry.

Mathieu Aubry received MSc degree from the Ecole Polytechnique, PhD from the Ecole Normale Superieure and Habilitation from Universite Paris-Est. In 2015, he spent a year as a visiting researcher at UC Berkeley and is a researcher at Ecole des Ponts. He works on 3D shape analysis and developing Computer Vision approaches for Digital Humanities.

Josef Sivic received MSc degree from the Czech Technical University in Prague, PhD from the University of Oxford and Habilitation from Ecole Normale Superieure in Paris. He currently holds a senior researcher position at Inria in Paris and a Distinguished Senior Researcher position at the Czech Institute of Robotics, Informatics and Cybernetics at the Czech Technical University in Prague where he leads a newly created team on Intelligent Machine Perception. He has published more than 60 scientific publications and his papers have been awarded the Longuet-Higgins prize (CVPR'07) and the Helmholtz prize (ICCV'03 and ICCV'05) for fundamental contributions to computer vision. He has served as an area chair for major computer vision conferences and as a program chair for ICCV'15. In 2013, he has received an ERC starting grant.