Fast Bi-Layer Neural Synthesis of One-Shot Realistic Head Avatars

Egor Zakharov[1,2], Aleksei Ivakhnenko[1], Aliaksandra Shysheya[1,3], and Victor Lempitsky[1,2(✉)]

[1] Samsung AI Center – Moscow, Moscow, Russia
v.lempitsky@samsung.com
[2] Skolkovo Institute of Science and Technology, Moscow, Russia
[3] University of Cambridge, Cambridge, UK

Abstract. We propose a neural rendering-based system that creates head avatars from a single photograph. Our approach models a person's appearance by decomposing it into two layers. The first layer is a pose-dependent coarse image that is synthesized by a small neural network. The second layer is defined by a pose-independent texture image that contains high-frequency details. The texture image is generated offline, warped and added to the coarse image to ensure a high effective resolution of synthesized head views. We compare our system to analogous state-of-the-art systems in terms of visual quality and speed. The experiments show significant inference speedup over previous neural head avatar models for a given visual quality. We also report on a real-time smartphone-based implementation of our system.

Keywords: Neural avatars · Talking heads · Neural rendering · Head synthesis · Head animation

1 Introduction

Personalized head avatars driven by keypoints or other mimics/pose representation is a technology with manifold applications in telepresence, gaming, AR/VR applications, and special effects industry. Modeling human head appearance is a daunting task, due to complex geometric and photometric properties of human heads including hair, mouth cavity and surrounding clothing. For at least two decades, creating head avatars (talking head models) was done with computer graphics tools using mesh-based surface models and texture maps. The resulting systems fall into two groups. Some [4] are able to model specific people with very high realism after significant acquisition and design efforts are spent on those particular people. Others [18] are able to create talking head models from as little as a single photograph, but do not aim to achieve photorealism.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58610-2_31) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12357, pp. 524–540, 2020.
https://doi.org/10.1007/978-3-030-58610-2_31

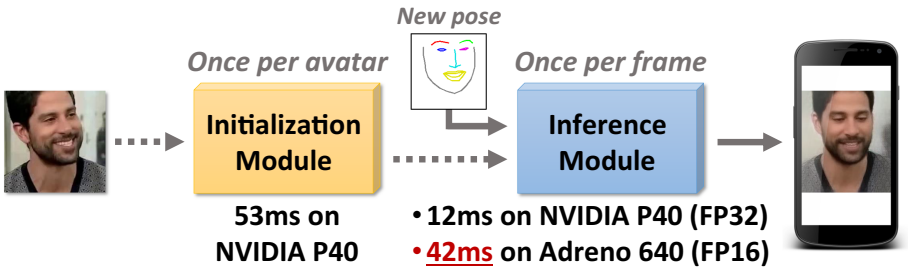

Fig. 1. Our new architecture creates photorealistic neural avatars in one-shot mode and achieves considerable speed-up over previous approaches. Rendering takes just 42 milliseconds on Adreno 640 (Snapdragon 855) GPU, FP16 mode.

In recent years, *neural talking heads* have emerged as an alternative to classic computer graphics pipeline striving to achieve both high realism and ease of acquisition. The first works required a video [25,38] or even multiple videos [27,34] to create a neural network that can synthesize talking head view of a person. Most recently, several works [12,16,32,32,35,36,40] presented systems that create neural head avatars from a handful of photographs (*few-shot* setting) or a single photograph (*one-shot* setting), causing both excitement and concerns about potential misuse of such technology.

Existing few-shot neural head avatar systems achieve remarkable results. Yet, unlike some of the graphics-based avatars, the neural systems are too slow to be deployed on mobile devices and require a high-end desktop GPU to run in real-time. We note that most application scenarios of neural avatars, especially those related to telepresence, would benefit highly from the capability to run in real-time on a mobile device. While in theory neural architectures within state-of-the-art approaches can be scaled down in order to run faster, we show that such scaling down results in a very unfavourable speed-realism tradeoff.

In this work, we address the speed limitations of one-shot neural head avatar systems, and develop an approach that can run much faster than previous models. To achieve this, we adopt a *bi-layer representation*, where the image of an avatar in a new pose is generated by summing two components: a coarse image directly predicted by a rendering network, and a warped texture image. While the warping itself is also predicted by the rendering network, the texture is estimated at the time of avatar creation and is static at runtime. To enable the few-shot capability, we use the meta-learning stage on a dataset of videos, where we (meta)-train the inference (rendering) network, the embedding network, as well as the texture generation network.

The separation of the target frames into two layers allows us both to improve the effective resolution and the speed of neural rendering. This is because we can use off-line avatar generation stage to synthesize high-resolution texture, while at test time both the first component (coarse image) and the warping of the texture need not contain high frequency details and can therefore be predicted by a relatively small rendering network. These advantages of our system are validated

by extensive comparisons with previously proposed neural avatar systems. We also report on the smartphone-based real-time implementation of our system, which was beyond the reach of previously proposed models.

2 Related Work

As discussed above, methods for the neural synthesis of realistic talking head sequences can be divided into many-shot (i.e. requiring a video or multiple videos of the target person for learning the model) [20,25,27,38] and a more recent group of few-shot/singe-shot methods capable of acquiring the model of a person from a single or a handful photographs [16,32,35,36,39,40]. Our method falls into the latter category as we focus on the one-shot scenario (modeling from a single photograph).

Along another dimension, these methods can be divided according to the architecture of the generator network. Thus, several methods [25,35,38,40] use generators based on *direct synthesis*, where the image is generated using a sequence of convolutional operators, interleaved with elementwise non-linearities, and normalizations. Person identity information may be injected into such architecture, either with a lengthy learning process (in the many-shot scenario) [25,38] or by using adaptive normalizations conditioned on person embeddings [12,35,40]. The method [40] effectively combines both approaches by injecting identity through adaptive normalizations, and then fine-tuning the resulting generator on the few-shot learning set. The direct synthesis approach for human heads can be traced back to [34] that generated lips of a famous person in the talking head sequence, and further towards first works on conditional convolutional neural synthesis of generic objects such as [10].

The alternative to the direct image synthesis is to use differentiable warping [21] inside the architecture. The X2Face approach [39] applies warping twice, first from the source image to a standardized image (texture), and then to the target image. The Codec Avatar system [27] synthesizes a pose-dependent texture for a simplified mesh geometry. The MarioNETte system [16] applies warping to the intermediate feature representations. The Few-shot Vid-to-Vid system [36] combines direct synthesis with the warping of the previous frame in order to obtain temporal continuity. The First Order Motion Model [32] learns to warp the intermediate feature representation of the generator based on keypoints that are learned from data. Beyond heads, differentiable warping/texturing have recently been used for full body re-rendering [29,31]. Earlier, DeepWarp system [13] used neural warping to alter the appearance of eyes for the purpose of gaze redirection, and [42] also used neural warping for the resynthesis of generic scenes. Our method combines direct image synthesis with warping in a new way, as we obtain the fine layer by warping an RGB *pose-independent* texture, while the coarse-grained *pose-dependent* RGB component is synthesized by a neural network directly.

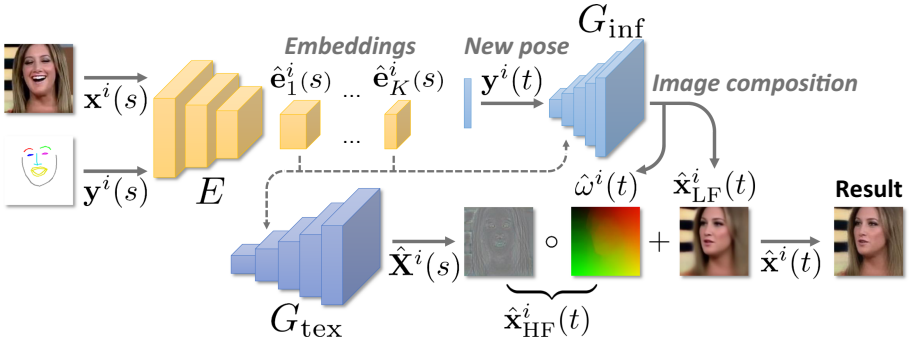

Fig. 2. During training, we first encode a source frame into the embeddings, then we initialize adaptive parameters of both inference and texture generators, and predict a high-frequency texture. These operations are only done once per avatar. Target keypoints are then used to predict a low-frequency component of the output image and a warping field, which, applied to the texture, provides the high-frequency component. Two components are then added together to produce an output.

3 Methods

We use video sequences annotated with keypoints and, optionally, segmentation masks, for training. We denote t-th frame of the i-th video sequence as $\mathbf{x}^i(t)$, corresponding keypoints as $\mathbf{y}^i(t)$, and segmentation masks as $\mathbf{m}^i(t)$ We will use an index t to denote a target frame, and s – a source frame. Also, we mark all tensors, related to generated images, with a hat symbol, ex. $\hat{\mathbf{x}}^i(t)$. We assume the spatial size of all frames to be constant and denote it as $H \times W$. In some modules, input keypoints are encoded as an RGB image, which is a standard approach in a large body of previous works [16,36,40]. In this work, we will call it a *landmark* image. But, contrary to these approaches, at test-time we input the keypoints into the inference generator directly as a vector. This allows us to significantly reduce the inference time of the method.

3.1 Architecture

In our approach, the following networks are trained in an end-to-end fashion:

- The *embedder* network $E\big(\mathbf{x}^i(s), \mathbf{y}^i(s)\big)$ encodes a concatenation of a source image and a landmark image into a stack of embeddings $\{\hat{\mathbf{e}}^i_k(s)\}$, which are used for initialization of the adaptive parameters inside the generators.
- The *texture generator* network $G_{\text{tex}}\big(\{\hat{\mathbf{e}}^i_k(s)\}\big)$ initializes its adaptive parameters from the embeddings and decodes an inpainted high-frequency component of the source image, which we call a texture $\hat{\mathbf{X}}^i(s)$.
- The *inference generator* network $G\big(\mathbf{y}^i(t), \{\hat{\mathbf{e}}^i_k(s)\}\big)$ maps target poses into a predicted image $\hat{\mathbf{x}}^i(t)$. The network accepts vector keypoints as an input and outputs a low-frequency layer of the output image $\hat{\mathbf{x}}^i_{\text{LF}}(t)$, which encodes

basic facial features, skin color and lighting, and $\hat{\omega}^i(t)$ – a mapping between coordinate spaces of the texture and the output image. Then, the high-frequency layer of the output image is obtained by warping the predicted texture: $\hat{\mathbf{x}}^i_{\mathrm{HF}}(t) = \hat{\omega}^i(t) \circ \hat{\mathbf{X}}^i(s)$, and is added to a low-frequency component to produce the final image:

$$\hat{\mathbf{x}}^i(t) = \hat{\mathbf{x}}^i_{\mathrm{LF}}(t) + \hat{\mathbf{x}}^i_{\mathrm{HF}}(t)\,. \tag{1}$$

– Finally, the *discriminator* network $D\big(\mathbf{x}^i(t), \mathbf{y}^i(t)\big)$, which is a conditional [28] relativistic [23] PatchGAN [20], maps a real or a synthesised target image, concatenated with the target landmark image, into realism scores $\mathbf{s}^i(t)$.

During training, we first input a source image $\mathbf{x}^i(s)$ and a source pose $\mathbf{y}^i(s)$, encoded as a landmark image, into the embedder. The outputs of the embedder are K tensors $\hat{\mathbf{e}}^i_k(s)$, which are used to predict the adaptive parameters of the texture generator and the inference generator. A high-frequency texture $\hat{\mathbf{X}}^i(s)$ of the source image is then synthesized by the texture generator. Next, we input corresponding target keypoints $\mathbf{y}^i(t)$ into the inference generator, which predicts a low-frequency component of the output image $\hat{\mathbf{x}}^i_{\mathrm{LF}}(t)$ directly and a high-frequency component $\hat{\mathbf{x}}^i_{\mathrm{HF}}(t)$ by warping the texture with a predicted field $\hat{\omega}^i(t)$. Finally, the output image $\hat{\mathbf{x}}^i(t)$ is obtained as a sum of these two components.

It is important to note that while the texture generator is manually forced to generate only a high-frequency component of the image via the design of the loss functions, which is described in the next section, we do not specifically constrain it to perform texture inpainting for occluded head parts. This behavior is emergent from the fact that we use two different images with different poses for initialization and loss calculation.

3.2 Training Process

We use multiple loss functions for training. The main loss function responsible for the realism of the outputs is trained in an adversarial way [15]. We also use pixelwise loss to preserve source lightning conditions and perceptual [22] loss to match the source identity in the outputs. Finally, a regularization of the texture mapping adds robustness to the random initialization of the model.

Pixelwise and Perceptual Losses ensure that the predicted images match the ground truth, and are respectively applied to low- and high-frequency components of the output images. Since usage of pixelwise losses assumes independence of all pixels in the image, the optimization process leads to blurry images [20], which is suitable for the low-frequency component of the output. Thus the pixelwise loss is calculated by simply measuring mean L_1 distance between the target image and the low-frequency component:

$$\mathcal{L}^G_{\mathrm{pix}} = \frac{1}{HW} ||\hat{\mathbf{x}}^i_{\mathrm{LF}}(t) - \mathbf{x}^i(t)||_1\,. \tag{2}$$

On the contrary, the optimization of the perceptual loss leads to crisper and more realistic images [22], which we utilize to train the high-frequency component. To calculate the perceptual loss, we use the stop-gradient operator SG, which allows us to prevent the gradient flow into a low-frequency component. The input generated image is, therefore, calculated as following:

$$\tilde{\mathbf{x}}^i(t) = \mathrm{SG}\big(\hat{\mathbf{x}}^i_{\mathrm{LF}}(t)\big) + \hat{\mathbf{x}}^i_{\mathrm{HF}}(t). \tag{3}$$

Following [16] and [40], our variant of the perceptual loss consists of two components: features evaluated using an ILSVRC (ImageNet) pre-trained VGG19 network [33], and the VGGFace network [30], trained for face recognition. If we denote the intermediate features of these networks as $\mathbf{f}^i_{k,\mathrm{IN}}(t)$ and $\mathbf{f}^i_{k,\mathrm{face}}(t)$, and their spatial size as $H_k \times W_k$, the objectives can be written as follows:

$$\mathcal{L}^G_{\mathrm{IN}} = \frac{1}{K}\sum_k \frac{1}{H_k W_k} ||\tilde{\mathbf{f}}^i_{k,\mathrm{IN}}(t) - \mathbf{f}^i_{k,\mathrm{IN}}(t)||_1, \tag{4}$$

$$\mathcal{L}^G_{\mathrm{face}} = \frac{1}{K}\sum_k \frac{1}{H_k W_k} ||\tilde{\mathbf{f}}^i_{k,\mathrm{face}}(t) - \mathbf{f}^i_{k,\mathrm{face}}(t)||_1. \tag{5}$$

Texture Mapping Regularization is proposed to improve the stability of the training. In our model, the coordinate space of the texture is learned implicitly, and there are two degrees of freedom that can mutually compensate each other: the position of the face in the texture, and the predicted warping. If, after initial iterations, the major part of the texture is left unused by the model, it can easily compensate that with a more distorted warping field. This artifact of an initialization is not fixed during training, and clearly is not the behavior we need, since we want all the texture to be used to achieve the maximum effective resolution in the outputs. We address the problem by regularizing the warping in the first iterations to be close to an identity mapping:

$$\mathcal{L}^G_{\mathrm{reg}} = \frac{1}{HW} ||\omega^i(t) - \mathcal{I}||_1 \,. \tag{6}$$

Adversarial Loss is optimized by both generators, the embedder and the discriminator networks. Usually, it resembles a binary classification loss function between real and fake images, which discriminator is optimized to minimize, and generators – maximize [15]. We follow a large body of previous works [6,16,36,40] and use a hinge loss as a substitute for the original binary cross entropy loss. We also perform relativistic realism score calculation [23], following its recent success in tasks such as super-resolution [38] and denoising [24]. Additionally, we use PatchGAN [20] formulation of the adversarial learning. The discriminator is trained only with respect to its adversarial loss $\mathcal{L}^D_{\mathrm{adv}}$, while the generators and the embedder are trained via the adversarial loss $\mathcal{L}^G_{\mathrm{adv}}$, and also a feature matching loss $\mathcal{L}_{\mathrm{FM}}$ [37]. The latter is introduced for better stability of the training.

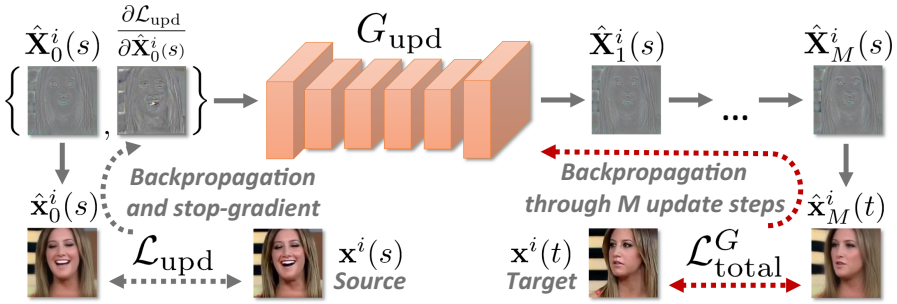

Fig. 3. Texture enhancement network (updater) accepts the current state of the texture and the guiding gradients to produce the next state. The guiding gradients are obtained by reconstructing the source image from the current state of the texture and matching it to the ground-truth via a lightweight updater loss. These gradients are only used as inputs and are detached from the computational graph. This process is repeated M times. The final state of the texture is then used to obtain a target image, which is matched to the ground-truth via the same loss as the one used during training of the main model. The gradients from this loss are then backpropagated through all M copies of the updater network.

3.3 Texture Enhancement

To minimize the identity gap, [40] suggested to fine-tune the generator weights to the few-shot training set. Training on a person-specific source data leads to significant improvement in realism and identity preservation of the synthesized images [40], but is computationally expensive. Moreover, when the source data is scarce, like in one-shot scenario, fine-tuning may lead to over-fitting and performance degradation, which is observed in [40]. We address both of these problems by using a learned gradient descend (LGD) method [5] to optimize only the synthesized texture $\hat{\mathbf{X}}^i(s)$. Optimizing with respect to the texture tensor prevents the model from overfitting, while the LGD allows us to perform optimization with respect to computationally expensive objectives by doing forward passes through a pre-trained network.

Specifically, we introduce a lightweight loss function $\mathcal{L}_{\text{upd}}$ (we use a sum of squared errors), that measures the distance between a generated image and a ground-truth in the pixel space, and a *texture updating* network G_{upd}, that uses the current state of the texture and the gradient of $\mathcal{L}_{\text{upd}}$ with respect to the texture to produce an update $\Delta\hat{\mathbf{X}}^i(s)$. During fine-tuning we perform M update steps, each time measuring the gradients of $\mathcal{L}_{\text{upd}}$ with respect to an updated texture. The visualization of the process can be seen in Fig. 3. More formally, each update is computed as:

$$\hat{\mathbf{X}}_{m+1}^i(s) = \hat{\mathbf{X}}_m^i(s) + G_{\text{upd}}\left(\hat{\mathbf{X}}_m^i(s), \frac{\partial \mathcal{L}_{\text{upd}}}{\partial \hat{\mathbf{X}}_{m(s)}^i}\right), \tag{7}$$

where $m \in \{0, \dots, M-1\}$ denotes an iteration number, with $\hat{\mathbf{X}}_0^i(s) \equiv \hat{\mathbf{X}}^i(s)$.

The network G_{upd} is trained by back-propagation through all M steps. For training, we use the same objective $\mathcal{L}^G_{\text{total}}$ that was used during the training of the base model. We evaluate it using a target frame $\mathbf{x}^i(t)$ and a generated frame

$$\hat{\mathbf{x}}^i_M(t) = \hat{\mathbf{x}}^i_{\text{LF}}(t) + \hat{\omega}^i(t) \circ \hat{\mathbf{X}}^i_M(s). \tag{8}$$

It is important to highlight that $\mathcal{L}_{\text{upd}}$ is not used for training of G_{upd}, but simply guides the updates to the texture. Also, the gradients with respect to this loss are evaluated using the source image, while the objective in Eq. 8 is calculated using the target image, which implies that the network has to produce updates for the whole texture, not just a region "visible" on the source image. Lastly, while we do not propagate any gradients into the generator part of the base model, we keep training the discriminator using the same objective $\mathcal{L}^D_{\text{adv}}$. Even though training the updater network jointly with the base generator is possible, and can lead to better quality (following the success of model agnostic meta-learning [11] method), we resort to two-stage training due to memory constraints.

3.4 Segmentation

The presence of static background leads to a certain degradation of our model for two reasons. Firstly, part of the capacity of the texture and the inference generators has to be spent on modeling high variety of background patterns. Secondly, and more importantly, the static nature of backgrounds in most training videos biases the warping towards identity mapping. We therefore, have found it advantageous to include background segmentation into our model.

We use a state-of-the-art face and body segmentation model [14] to obtain the ground truth masks. Then, we add the mask prediction output $\hat{\mathbf{m}}^i(t)$ to our inference generator alongside with its other outputs, and train it via a binary cross-entropy loss $\mathcal{L}_{\text{seg}}$ to match the ground truth mask $\mathbf{m}^i(t)$. To filter out the training signal, related to the background, we have explored multiple options. Simple masking of the gradients that are fed into the generator leads to severe overfitting of the discriminator. We also could not simply apply the ground truth masks to all the images in the dataset, since the model [14] works so well that it produces a sharp border between the foreground and the background, leading to border artifacts that emerge after adversarial training.

Instead, we have found out that masking the ground truth images that are fed to the discriminator with the predicted masks $\hat{\mathbf{m}}^i(t)$ works well. Indeed, these masks are smooth and prevent the discriminator from overfitting to the lack of background, or sharpness of the border. We do not backpropagate the signal from the discriminator and from perceptual losses to the generator via the mask pathway (i.e. we use stop gradient/detach operator $\text{SG}\big(\hat{\mathbf{m}}^i(t)\big)$ before applying the mask). The stop-gradient operator also ensures that the training does not converge to a degenerate state (empty foreground).

3.5 Implementation Details

All our networks consist of pre-activation residual blocks [17] with LeakyReLU activations. We set a minimum number of features in these blocks to 64, and a maximum to 512. By default, we use half the number of features in the inference generator, but we also evaluate our model with full- and quater-capacity inference part, with the results provided in the experiments section.

We use batch normalization [19] in all the networks except for the embedder and the texture updater. Inside the texture generator, we pair batch normalization with adaptive SPADE layers [36]. We modify these layers to predict pixel-wise scale and bias coefficients using feature maps, which are treated as model parameters, instead of being input from a different network. This allows us to save memory by removing additional networks and intermediate feature maps from the optimization process, and increase the batch size. Also, following [36], we predict weights for all 1×1 convolutions in the network from the embeddings $\{\hat{\mathbf{e}}^i_k(s)\}$, which includes the scale and bias mappings in AdaSPADE layers, and skip connections in the residual upsampling blocks. In the inference generator, we use standard adaptive batch normalization layers [6], but also predict weights for the skip connections from the embeddings.

We do simultaneous gradient descend on parameters of the generator networks and the discriminator using Adam [26] with a learning rate of $2 \cdot 10^{-4}$. We use 0.5 weight for adversarial losses, and 10 for all other losses, except for the VGGFace perceptual loss (Eq. 5), which is set to 0.01. The weight of the regularizer (Eq. 6) is then multiplicatively reduced by 0.9 every 50 iterations. We train our models on 8 NVIDIA P40 GPUs with the batch size of 48 for the base model, and a batch size of 32 for the updater model. We set unrolling depth M of the updater to 4 and use a sum of squared errors as the lightweight objective. Batch normalization statistics are synchronized across all GPUs during training. During inference they are replaced with "standing" statistics, similar to [6], which significantly improves the quality of the outputs, compared to the usage of running statistics. Spectral normalization is also applied in all linear and convolutional layers of all networks.

Please refer to the supplementary material for a detailed description of our model's architecture, as well as the discussion of training and architectural features that we have adopted.

4 Experiments

We perform evaluation in multiple scenarios. First, we use the original VoxCeleb2 [8] dataset to compare with state-of-the-art systems. To do that, we annotated this dataset using an off-the-shelf facial landmarks detector [7]. Overall, the dataset contains 140697 videos of 5994 different people. We also use a high-quality version of the same dataset, additionally annotated with the segmentation masks (which were obtained using a model [14]), to measure how the performance of our model scales with a dataset of a significantly higher quality.

We obtained this version by downloading the original videos via the links provided in the VoxCeleb2 dataset, and filtering out the ones with low resolution. This dataset is, therefore, significantly smaller and contains only 14859 videos of 4242 people, with each video having at most 250 frames (first 10 s). Lastly, we do ablation studies on both VoxCeleb2 and VoxCeleb2-HQ, and report on a smartphone-based implementation of the method. For comparisons and ablation studies we show the results qualitatively and also evaluate the following metrics:

- Learned perceptual image patch similarity [41] (LPIPS), which measures overall predicted image similarity to ground truth.
- Cosine similarity between the embedding vectors of a state-of-the-art face recognition network [9] (CSIM), calculated using the synthesized and the target images. This metric evaluates the identity mismatch.
- Normalized mean error of the head pose in the synthesized image (NME). We use the same network [7], which was used for the annotation of the dataset, to evaluate the pose of the synthesized image. We normalize the error, which is a mean euclidean distance between the predicted and the target points, by the distance between the eyes in the target pose, multiplied by 10.
- Multiply-accumulate operations (MACs), which measure the complexity of each method. We exclude from the evaluation initialization steps, which are calculated only once per avatar.

The test set in both datasets does not intersect with the train set in terms of videos or identities. For evaluation, we use a subset of 50 test videos with different identities (for VoxCeleb2, it is the same as in [40]). The first frame in each sequence is used as a source. Target frames are taken sequentially at 1 FPS.

We only discuss most important results in the main paper. For additional qualitative results and comparisons please refer to the supplementary materials.

4.1 Comparison with the State-of-the-art Methods

We compare against three state-of-the-art systems: Few-shot Talking Heads [40], Few-shot Vid-to-Vid [36] and First Order Motion Model [32]. The first system is a problem-specific model designed for avatar creation. Few-shot Vid-to-Vid is a state-of-the-art video-to-video translation system, which has also been successfully applied to this problem. First Order Motion Model (FOMM) is a general motion transfer system that does not use precomputed keypoints, but can also be used as an avatar. We believe that these models are representative of the most recent and successful approaches to one-shot avatar generation. We also acknowledge the work of [16], but do not compare to them extensively due to unavailability of the source code, pretrained models or pre-calculated results. A small-scale qualitative comparison is provided in the supplementary materials. Additionally, their method is limited to the usage of 3D keypoints, while our method does not have such restriction. Lastly, since Few-shot Vid-to-Vid is an autoregressive model, we use a full test video sequence for evaluation (25 FPS) and save the predicted frames at 1 FPS.

Fig. 4. In order to evaluate a quality against performance trade off, we train a family of models with varying complexity for each of the compared methods. For quality metrics, we have compared synthesized images to their targets using a perceptual image similarity (LPIPS ↓), identity preservation metric (CSIM ↑), and a normalized pose error (NME ↓). We highlight a model which was used for the comparison in Fig. 5 with a bold marker. We observe that our model outperforms the competitors in terms of identity preservation (CSIM) and pose matching (NME) in the settings, when models' complexities are comparable. In order to better compare with FOMM, we did a user study, where users have preferred the image generated by our model to FOMM 59.6% of the time.

Importantly, the base models in these approaches have a lot of computational complexity, so for each method we evaluate a family of models by varying the number of parameters. The performance comparison for each family is reported in Fig. 4 (with Few-shot Talking Heads being excluded from this evaluation, since their performance is much worse than the compared methods). Overall, we can see that our model's family outperforms competing methods in terms of pose error and identity preservation, while being, on average, up to an order of magnitude faster. To better compare with FOMM in terms of image similarity, we have performed a user study, where we asked crowd-sourced users which generated image better matches the ground truth. In total, 361 users evaluated 1600 test pairs of images, with each one seeing on average 21 pairs. In 59.6% of comparisons, the result of our medium model was preferred to a medium sized model of FOMM.

Another important note is on how the complexity was evaluated. In Few-shot Vid-to-Vid we have additionally excluded from the evaluation parts that are responsible for the temporal consistency, since other compared methods are evaluated frame-by-frame and do not have such overhead. Also, in FOMM we have excluded the keypoints extractor network, because this overhead is shared implicitly by all the methods via usage of the precomputed keypoints.

We visualize the results for medium-sized models of each of the compared methods in Fig. 5. Since all methods perform similarly in case when source and target images have marginal differences, we have shown the results where a source and a target have different head poses. In this extrapolation setting, our method has a clear advantage, while other methods either introduce more artifacts or more blurriness.

Fig. 5. Comparison on a VoxCeleb2 dataset. The task is to reenact a **target** image, given a **source** image and target keypoints. The compared methods are Few-shot Talking Heads [40], Few-shot Vid-to-Vid [36], First Order Motion Model (FOMM) [32] and our proposed Bi-layer Model. For each method, we used the models with a similar number of parameters, and picked source and target images to have diverse poses and expressions, in order to highlight the differences between the compared methods.

Evaluation on High-Quality Images. Next, we evaluate our method on the high-quality dataset and present the results in Fig. 6. Overall, in this case, our method is able to achieve a smaller identity gap, compared to the dataset with the background. We also show the decomposition between the texture and a low frequency component in Fig. 7. Lastly, in Fig. 8, we show that our texture enhancement pipeline allows us to render small person-specific features like wrinkles and moles on out-of-domain examples. For more qualitative examples, as well as reenactment examples with a driver of a different person, please refer to the supplementary materials.

Smartphone-Based Implementation. We train our model using PyTorch [1] and then port it to smartphones with Qualcomm Snapdragon chips. There are several frameworks which provide APIs for mobile inference on such devices. From our experiments, we measured the Snapdragon Neural Processing Engine (SNPE) [2] to be about 1.5 times faster than PyTorch Mobile [1] and up to two times faster than TensorFlow Lite [3]. The medium-sized model ported to the

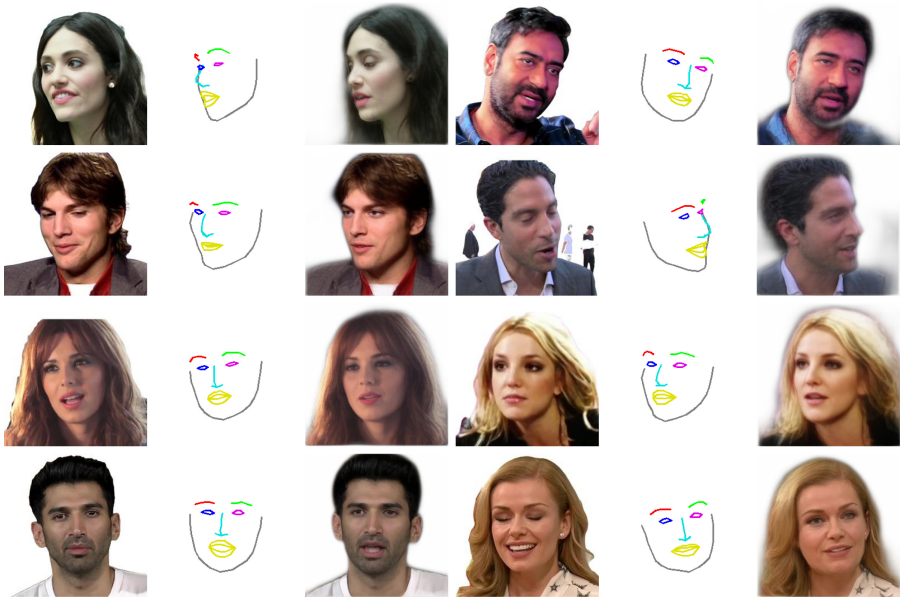

Fig. 6. High quality synthesis results. We can see that our model is both capable of viewpoint extrapolation and low identity gap synthesis. The architecture in this experiment has the same number of parameters as the medium architecture in the previous comparison.

Snapdragon 855 (Adreno 640 GPU, FP16 mode) takes 42 ms per frame, which is sufficient for real-time performance, given that the keypoint tracking is being run in parallel, e.g. on a mobile CPU.

Ablation Study. Finally, we evaluate the contribution of individual components. First, we evaluate the contribution of adaptive SPADE layers in the texture generator (by replacing them with adaptive batch normalization and per-pixel biases) and adaptive skip-connections in both generators. A model with these features removed makes up our baseline. Lastly, we evaluate the contribution of the updater network. The results can be seen in Table 1 and Fig. 9. We evaluate the baseline approach only on a VoxCeleb2 dataset, while the full models with and without the updater network are evaluated on both low- and high-quality datasets. Overall, we see a significant contribution of each component with respect to all metrics, which is particularly noticeable in the high-quality scenario. In all ablation comparisons, medium-sized models were used.

Fig. 7. Detailed results on the generation process of the output image. **LF** denotes a low-frequency component.

Fig. 8. Our method can preserve a lot of details in the facial features, like the famous Marylin's mole.

Table 1. Ablation studies of our approach. We first evaluate the baseline method without AdaSPADE or adaptive skip connections. Then we add these layers, following [36], and observe significant quality improvement. Finally, our updater network provides even more improvement across all metrics, especially noticeable in the high-quality scenario.

Method	LPIPS ↓	CSIM ↑	NME ↓
VoxCeleb2			
Baseline	0.377	0.547	0.447
Ours	0.370	0.595	0.441
+Updater	**0.358**	**0.653**	**0.433**
VoxCeleb2-HQ			
Ours	0.313	0.432	0.476
+Updater	**0.298**	**0.649**	**0.456**

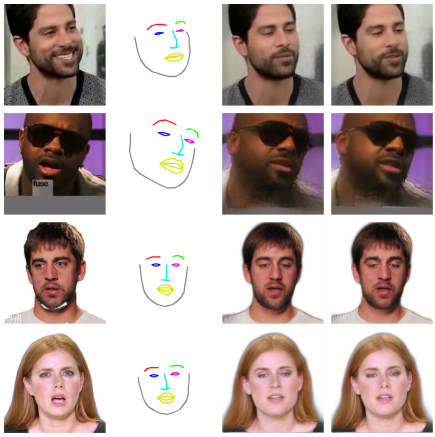

Fig. 9. Examples from the ablation study on VoxCeleb2 (first two rows) and VoxCeleb2-HQ (last two rows).

5 Conclusion

We have proposed a new neural rendering-based system that creates head avatars from a single photograph. Our approach models person appearance by decomposing it into two layers. The first layer is a pose-dependent coarse image that is synthesized by a small neural network. The second layer is defined by a pose-independent texture image that contains high-frequency details and is generated offline. During test-time it is warped and added to the coarse image to ensure high effective resolution of synthesized head views. We compare our system to analogous state-of-the-art systems in terms of visual quality and speed. The experiments show up to an order of magnitude inference speedup over

previous neural head avatar models, while achieving state-of-the-art quality. We also report on a real-time smartphone-based implementation of our system.

References

1. PyTorch homepage. https://pytorch.org
2. SNPE homepage. https://developer.qualcomm.com/sites/default/files/docs/snpe
3. TensorFlow Lite homepage. https://www.tensorflow.org/lite
4. Alexander, O., et al.: The Digital Emily project: achieving a photorealistic digital actor. IEEE Comput. Graph. Appl. **30**(4), 20–31 (2010)
5. Andrychowicz, M., et al.: Learning to learn by gradient descent by gradient descent. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (2016)
6. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: 7th International Conference on Learning Representations, ICLR 2019 (2019)
7. Bulat, A., Tzimiropoulos, G.: How far are we from solving the 2D & 3D face alignment problem? (and a dataset of 230, 000 3D facial landmarks). In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017, pp. 1021–1030 (2017)
8. Chung, J.S., Nagrani, A., Zisserman, A.: Voxceleb2: deep speaker recognition. In: Interspeech 2018, 19th Annual Conference of the International Speech Communication Association (2018)
9. Deng, J., Guo, J., Niannan, X., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: CVPR (2019)
10. Dosovitskiy, A., Tobias Springenberg, J., Brox, T.: Learning to generate chairs with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1538–1546 (2015)
11. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017 (2017)
12. Fu, C., Hu, Y., Wu, X., Wang, G., Zhang, Q., He, R.: High fidelity face manipulation with extreme pose and expression. arXiv preprint arXiv:1903.12003 (2019)
13. Ganin, Y., Kononenko, D., Sungatullina, D., Lempitsky, V.: DeepWarp: photorealistic image resynthesis for gaze manipulation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 311–326. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_20
14. Gong, K., Gao, Y., Liang, X., Shen, X., Wang, M., Lin, L.: Graphonomy: universal human parsing via graph transfer learning. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2019)
15. Goodfellow, I.J., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014 (2014)
16. Ha, S., Kersner, M., Kim, B., Seo, S., Kim, D.: Marionette: Few-shot face reenactment preserving identity of unseen targets. CoRR abs/1911.08139 (2019)
17. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38

18. Hu, L., et al.: Avatar digitization from a single image for real-time rendering. ACM Trans. Graph. **36**(6), 195:1–195:14 (2017)
19. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015 (2015)
20. Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 (2017)
21. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems (2015)
22. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 694–711. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_43
23. Jolicoeur-Martineau, A.: The relativistic discriminator: a key element missing from standard GAN. In: 7th International Conference on Learning Representations, ICLR (2019)
24. Kim, D., Chung, J.R., Jung, S.: GRDN: grouped residual dense network for real image denoising and GAN-based real-world noise modeling. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019 (2019)
25. Kim, H., et al.: Deep video portraits. arXiv preprint arXiv:1805.11714 (2018)
26. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2014)
27. Lombardi, S., Saragih, J., Simon, T., Sheikh, Y.: Deep appearance models for face rendering. ACM Trans. Graph. (TOG) **37**(4), 68 (2018)
28. Mirza, M., Osindero, S.: Conditional generative adversarial nets. CoRR abs/1411.1784 (2014)
29. Neverova, N., Alp Güler, R., Kokkinos, I.: Dense pose transfer. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11207, pp. 128–143. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01219-9_8
30. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: Proceedings of the British Machine Vision Conference 2015, BMVC 2015 (2015)
31. Shysheya, A., et al.: Textured neural avatars. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2019)
32. Siarohin, A., Lathuilière, S., Tulyakov, S., Ricci, E., Sebe, N.: First order motion model for image animation. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019 (2019)
33. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014). http://arxiv.org/abs/1409.1556
34. Suwajanakorn, S., Seitz, S.M., Kemelmacher-Shlizerman, I.: Synthesizing Obama: learning lip sync from audio. ACM Trans. Graph. (TOG) **36**(4), 95 (2017)
35. Tripathy, S., Kannala, J., Rahtu, E.: ICface: interpretable and controllable face reenactment using GANs. CoRR abs/1904.01909 (2019). http://arxiv.org/abs/1904.01909
36. Wang, T., Liu, M., Tao, A., Liu, G., Catanzaro, B., Kautz, J.: Few-shot video-to-video synthesis. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019 (2019)

37. Wang, T., Liu, M., Zhu, J., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional GANs. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018 (2018)
38. Wang, T., et al.: Video-to-video synthesis. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3–8 December 2018, Montréal, Canada (2018)
39. Wiles, O., Koepke, A.S., Zisserman, A.: X2Face: a network for controlling face generation using images, audio, and pose codes. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11217, pp. 690–706. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01261-8_41
40. Zakharov, E., Shysheya, A., Burkov, E., Lempitsky, V.S.: Few-shot adversarial learning of realistic neural talking head models. In: IEEE International Conference on Computer Vision, ICCV 2019 (2019)
41. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2018)
42. Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A.: View synthesis by appearance flow. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 286–301. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_18