ONE BASIS TO ANIMATE THEM ALL: GAUSSIAN BLENDSHAPE DISTILLATION FOR REAL-TIME AVATARS

Anonymous authors
Paper under double-blind review

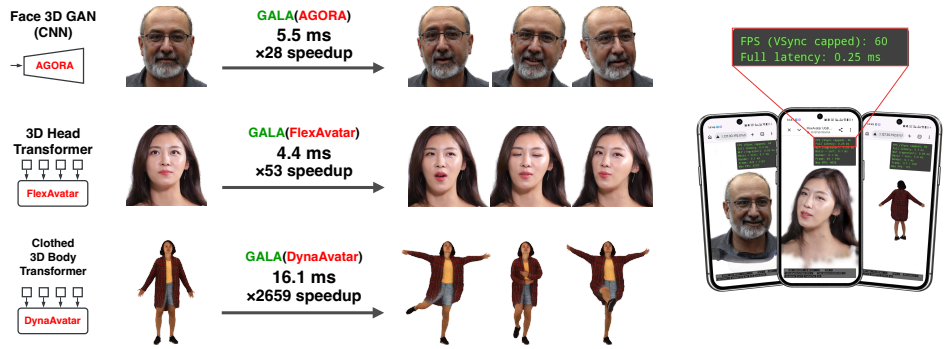

Figure 1: GALA enables efficient, high-fidelity animation across diverse Gaussian avatar models. By introducing shared blendshapes and a lightweight coefficient predictor, GALA accelerates CPU animation by up to three orders of magnitude and enables real-time deployment on mobile devices.

ABSTRACT

3D Gaussian avatars support fast rendering, however, their real-time animation is often challenged by the costly neural inference. We address this bottleneck and show that the animation of pretrained avatar models can be closely approximated by a linear combination of identity-independent blendshapes. Building on this finding, we introduce **GALA** (**G**aussian **A**nimation via **L**inear **A**pproximation), a distillation method that replaces per-frame heavy neural decoding with a shallow coefficient predictor and a linear blend. To improve fidelity and reduce memory requirements, we propose to construct the basis using block-local PCA under a rendering-aware metric and a memory budget. Our method learns a shallow MLP network to predict blendshape coefficients and applies to various animation architectures without retraining original models. We validate GALA by accelerating the inference of three distinct avatar models for 3D animation of facial expressions and full-bodies with clothing dynamics. Across these models, our distillation generalizes to held-out identities and reduces CPU animation cost by up to three orders of magnitude while preserving most of the rendering quality. Excellent results of our method confirm the shared linear structure of learned avatar representations and enable highly efficient and accurate animation at frame rates reaching up to 60fps on mobile devices. *Our code and models will become publicly available.*

1 INTRODUCTION

Recent advances in neural avatars have substantially improved the generation, reconstruction, and animation of photorealistic digital humans. Representations based on 3D Gaussians are particularly attractive because their explicit primitives support efficient rendering (Kerbl et al., 2023), and they underlie models for head generation, reconstruction from a single image, and full-body animation with clothing dynamics (Yu et al., 2025; Fazylov et al., 2025; Kirschstein et al., 2026; Kwon et al., 2026). However, efficient rendering does not imply efficient animation. Many models evaluate a substantial neural network at every frame to update Gaussian attributes from expression or pose parameters, and this per-frame computation, which we call the model's *animation path*, can dominate inference cost and limits deployment on CPUs and mobile devices.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Comparison of avatar animation representations. Our method extracts an identity-shared blendshape basis from a pretrained nonlinear avatar model, complementing approaches based on predefined rigs or per-subject fitting. The extracted basis closely approximates the model's learned deformations and supports efficient linear blending for unseen identities.

	Photo-real.	Identity-shared	Linear at run time	Where the basis comes from
3D morphable models (Li et al., 2017b)	×	✓	✓	hand-built from scans
Per-person blendshapes (Zielonka et al., 2025)	✓	×	✓	fitted per subject
Fixed-rig avatars (He et al., 2025)	✓	✓	✓	chosen up front
Animatable avatar models (Yu et al., 2025)	✓	–	×	none (a network)
GALA (Ours)	✓	✓	✓	**discovered in a trained model**

The cost of neural animation raises the question of how much of the computation performed by a pretrained avatar model is necessary at inference time. The complexity of its nonlinear decoder does not establish that its outputs require an equally complex representation, and deformations across identities may share a compact linear structure even when the architecture and the training objective do not impose one. In this work, we show that the animation outputs of diverse pretrained Gaussian avatar models can be closely approximated by an identity-specific neutral avatar and a linear combination of identity-independent *blendshapes*, which we obtain by analyzing Gaussian attribute residuals relative to the neutral state across identities and driving signals. Crucially, a basis extracted once from a model also captures the animation of identities excluded from basis construction, since identity-dependent motion is retained through the blendshape coefficients while the deformation basis is shared.

Linear representations have a long history in face and body modeling (Blanz & Vetter, 1999; Li et al., 2017b). More recently, Gaussian blendshape methods have demonstrated efficient photorealistic animation with bases fitted to individual subjects (Ma et al., 2024; Zielonka et al., 2025), while systems such as LAM adopt a predefined animation rig (He et al., 2025). Our work investigates a different setting, extracting a reusable basis from a trained model that represents many identities (Table 1). The basis approximates deformations learned by the original network and can be reused for new subjects without reconstructing it, where identity independence refers to sharing within one pretrained model.

Building on this finding, we introduce a Gaussian blendshape distillation strategy, which we call **GALA** (**G**aussian **A**nimation via **L**inear **A**pproximation). Our method constructs separate bases for Gaussian attribute groups and spatial regions using block-local principal component analysis under a rendering-aware metric, which prioritizes deformations that matter for visual fidelity, and allocates components within a memory budget according to their estimated image-space error reduction per stored byte. A shallow coefficient predictor, conditioned on identity and driving signals, then learns to control the extracted basis, so that animation requires only coefficient prediction and a linear blend added to the cached neutral avatar. The procedure requires no retraining of the original model.

We apply our approach to three existing models with distinct architectures and training objectives, the *hosts* of our distillation: AGORA (Fazylov et al., 2025) for head generation, FlexAvatar (Kirschstein et al., 2026) for one-shot head reconstruction, and DynaAvatar (Kwon et al., 2026) for full-body reconstruction with clothing dynamics. GALA closely approaches animation accuracy of approximated host models and consistently outperforms other baselines on respective benchmarks. Crucially, our method increases CPU animation speed of host models by up to *three orders of magnitude* and enables on-device photorealistic 3D animation at 60 fps. Figure 1 illustrates qualitative results of our method. In summary, our contributions are threefold:

1. **Identity-independent linearization of neural animation.** We demonstrate that animation produced by complex pretrained Gaussian avatar networks can be closely approximated using a shared linear blendshape basis that generalizes to identities unseen during extraction.
2. **A method for basis construction and animation control.** We introduce GALA, combining rendering-aware, block-local basis construction under a memory budget with a shallow network that predicts blendshape coefficients from the identity and the driving signals.
3. **Extensive experimental validation.** We apply our method to three existing models for head and full-body animation and demonstrate significant improvements when compared to the state of the art and compact neural baselines.

2 Related Work

Animatable avatar models. Recent work on photorealistic avatars explores several approaches. 3D-aware GANs trained from unstructured images collections generate heads as triplanes (Chan et al., 2022) or Gaussians (Kirschstein et al., 2024; Hyun & Heo, 2024; Barthel et al., 2025), and their animatable versions condition generation on the parameters of morphable models through deformation fields or expression branches (Sun et al., 2023; Wu et al., 2022; Xu et al., 2023; Bergman et al., 2022; Yu et al., 2025; Fazylov et al., 2025). One-shot reconstruction methods regress an animatable head from a single image, either by using 3D GANs during training (Deng et al., 2024a;b; Tran et al., 2024) or by direct supervised reconstruction from tracked video (Chu et al., 2024; Chu & Harada, 2024; He et al., 2025; Kirschstein et al., 2026), and diffusion priors pursue the same goal with synthesised supervision (Taubner et al., 2025b;a). Per-subject methods optimise Gaussian avatars from videos of particular people (Qian et al., 2024; Giebenhain et al., 2024; Chen et al., 2024). Work on full-body animation follows similar approaches. One-shot full-body reconstruction methods (Qiu et al., 2025; Zhuang et al., 2025; Sim & Moon, 2025) articulate Gaussians by SMPL-X skinning (Pavlakos et al., 2019), while DynaAvatar (Kwon et al., 2026) adds pose-dependent clothing dynamics using a transformer conditioned on pose history. We apply our approach to two recent host models for head animation AGORA (Fazylov et al., 2025), FlexAvatar (Kirschstein et al., 2026) and one host model for full-body animation DynaAvatar (Kwon et al., 2026), and demonstrate significant improvement of their inference speed.

Linear face and body models. Linear representations have a long history in modeling human shape and appearance. Eigenfaces represent facial images in a low-dimensional PCA subspace (Sirovich & Kirby, 1987; Turk & Pentland, 1991), while active appearance models capture variations in shape and texture (Cootes et al., 2001). Three-dimensional morphable models extend this principle to facial geometry and appearance learned from scans (Blanz & Vetter, 1999). For animation, blendshape models express deformations as weighted combinations of offsets from a neutral shape (Parke, 1972; Lewis et al., 2014), and multilinear models disentangle identity and expression variation (Vlasic et al., 2005; Cao et al., 2014). Modern parametric models, including FLAME (Li et al., 2017b) and SMPL-X (Pavlakos et al., 2019), combine learned shape and corrective blendshapes with articulated skinning to represent expressive faces and bodies; see Egger et al. (2020) for a review of 3D morphable face models. These representations provide the driving parameters for the avatar models considered in our work. We show that the resulting neural animation outputs themselves admit a compact linear approximation: an identity-shared blendshape basis over Gaussian attributes, extracted from a pretrained network without direct fitting to scans.

Photorealistic blendshape models. Recent work extends blendshape representations to photorealistic Gaussian avatars. Methods such as 3D Gaussian Blendshapes (Ma et al., 2024), HeadGaS (Dhamo et al., 2024), Gaussian Déjà-vu (Yan et al., 2025), RGBAvatar (Li et al., 2025), GEM (Zielonka et al., 2025) and WebAvatar (Zhan et al., 2026) learn *subject-specific* bases over Gaussian attributes or latent features from monocular or multi-view observations. RGBAvatar predicts blending coefficients from tracked expression parameters, while GEM extracts a basis from a trained subject-specific avatar network and predicts its coefficients from images. Similarly, SqueezeMe (Iandola et al., 2025) distils per-subject full-body Gaussian avatars into linear corrective bases for efficient headset rendering. These methods establish the effectiveness of linear representations for individual avatars, but require a separate basis for each subject. Fixed-rig approaches such as LAM (He et al., 2025) instead support efficient animation through a predefined FLAME-based articulation model. Differently from previous work, our method extracts a shared identity-agnostic blendshape basis from a pretrained neural model. The basis captures deformations learned by that model and is reused unchanged for identities unseen during extraction, extending Gaussian blendshape distillation from individual subjects to models of diverse identities.

3 Method

Our goal is to accelerate pretrained Gaussian avatar models (*host models*) by replacing their per-frame neural decoding with a fast inference based on linear blendshape representations. We first introduce a generic host model and its animation parameters in Section 3.1. Section 3.2 then presents main parts of our method, namely the construction of identity-shared basis and prediction of its coefficients from animation parameters.

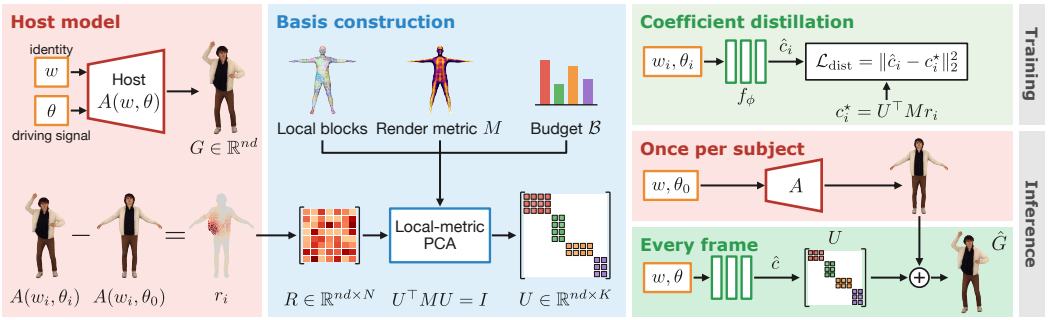

Figure 2: Overview of GALA. Residuals r_i of the host model A with respect to the neutral avatar $A(w_i, \theta_0)$ (Section 3.2.1) are decomposed by a spatially local, rendering-aware PCA under a memory budget $\mathcal{B}$ into the block-diagonal basis U (Section 3.2.2). A coefficient network f_ϕ is trained to regress the projected coefficients $c^\star$ (Section 3.2.3). At inference, the neutral avatar is computed once per subject, and every frame costs one evaluation of a shallow network f_ϕ and a linear blend.

3.1 Preliminaries

We consider avatar models that represent a subject using n 3D Gaussians rendered by differentiable splatting (Kerbl et al., 2023). Each Gaussian is described by a d-dimensional attribute vector, which includes position, scale, rotation, opacity, and appearance, with additional skinning weights where required. Avatar animation is controlled by parameters derived from parametric face or body models such as FLAME (Li et al., 2017b) or SMPL-X (Pavlakos et al., 2019). We denote the animation driving input by θ and its neutral setting by θ_0. For models with temporal conditioning, θ includes the required motion history. An identity descriptor w, either supplied directly or extracted from an input image, remains fixed throughout an animation sequence. We define pretrained host models as $A(w, \theta) \to G \in \mathbb{R}^{nd}$, where the output concatenates the attribute vectors of the n Gaussians. Importantly, Gaussian indices in the model output G must have consistent correspondence across variations in identity and animation parameters. This constraint is typically satisfied by existing models which initialize Gaussians on a shared UV grid or template-anchored point (Kirschstein et al., 2026; Yu et al., 2025; Fazylov et al., 2025; Kwon et al., 2026).

3.2 Gaussian Blendshape Distillation

In this section we present our method that aims to approximate the costly neural animation of the original host model $G = A(w, \theta)$ by a simpler model

$$\hat{G} = A(w, \theta_0) + Uc(w, \theta), \tag{1}$$

where $A(w, \theta_0)$ is the representation of a neutral avatar estimated only once per animation sequence, U is a linear basis and c is a vector of basis coefficients. The rest of this section explains our approach to construct the basis U and to efficiently estimate its coefficients $c(w, \theta)$. Figure 2 gives an overview of our method.

3.2.1 Simple PCA basis

Given a pretrained model $A(w, \theta)$ and the neutral pose θ_0, we compute residuals r for a large set of N samples corresponding to identities w_i and animation parameters θ_i as

$$r_i = A(w_i, \theta_i) - A(w_i, \theta_0). \tag{2}$$

We stack the residuals as columns of $R = [r_1, \ldots, r_N] \in \mathbb{R}^{nd\times N}$. A rank-$K$ basis can then be obtained by solving

$$\min_{U,\{c_i\}} \sum_{i=1}^{N} \|r_i - Uc_i\|_2^2, \qquad U^\top U = I. \tag{3}$$

By the Eckart–Young theorem (Eckart & Young, 1936), the solution is the truncated singular value decomposition $R \approx U\Sigma V^\top$, whose columns of U are the top K left singular vectors of R, with coefficients $c_i^\star = U^\top r_i$. We refer to this construction as uncentered PCA: residuals are measured relative to each identity's neutral avatar, and no additional mean offset is introduced. Consequently, zero coefficients recover the neutral avatar exactly.

Because residuals from different identities are factorized jointly, the resulting basis describes shared animation variation. It can also be applied to identities excluded from extraction, provided that Gaussian indices retain the correspondence specified in Section 3.1. However, this simple construction has three limitations. Attribute groups have different units and scales, global components incur storage costs even for localized deformations, and Euclidean attribute error does not directly reflect rendering quality. We address these limitations through the following modifications.

3.2.2 IMPROVED PCA BASIS

Per-attribute and spatially local decomposition. We first separate Gaussian attributes into groups a, each containing d_a channels, such as position, scale, rotation, opacity, and appearance. Constructing a separate basis for each group avoids coupling attributes with different scales and allows their representation capacities to be selected independently.

Since animation deformations are often spatially localized, a global basis wastes memory storing components over the entire avatar when only a small region contributes to the motion. Following this intuition, we partition the Gaussians into B spatial blocks using k-means such that the Gaussians within clusters share similar 3D locations. For attribute group a and block j, containing n_j Gaussians, we construct a basis

$$U_{a,j} \in \mathbb{R}^{n_j d_a \times K_{a,j}} \tag{4}$$

with $K_{a,j}$ components. The complete basis U is formed by assembling these local bases in block-diagonal form after reordering the attributes. Each local component stores $n_j d_a$ values instead of $n d_a$ and encodes a specific spatial region of the avatar. To limit the size of the increased basis U, we reduce the number of components $K_{a,j}$ in $U_{a,j}$ allowing the memory budget to be concentrated on regions with significant variations and complex motion, see below.

Rendering-aware PCA. Standard PCA allocates representation capacity to directions with the largest variation in Gaussian attribute space. To prioritize deformations that affect visual quality, we replace the Euclidean reconstruction objective with a rendering-aware quadratic metric. Let $I(x)$ denote an image rendered from Gaussian attributes x, and let $J = \partial I / \partial x$. For a small attribute perturbation δ,

$$\|I(x+\delta) - I(x)\|_2^2 \approx \delta^\top H \delta, \qquad H = J^\top J. \tag{5}$$

The full matrix H is too large to construct. We therefore estimate only its per-Gaussian diagonal blocks within each attribute group by stochastic probing (Hutchinson, 1989; Bekas et al., 2007) (Appendix A). The estimate for attribute group a, normalized to unit mean trace per channel, is denoted as $\widehat{H}_a$. Applying the same procedure to the landmark positions instead of the image gives $\widehat{H}_a^{\mathrm{lmk}}$. We define the metric of attribute group a as

$$M_a = \alpha I + \beta \widehat{H}_a + \gamma \widehat{H}_a^{\mathrm{lmk}}. \tag{6}$$

The term $\beta \widehat{H}_a$ weights attribute errors by their effect on the rendered image, the ridge term αI with $\alpha > 0$ retains sensitivity to directions that are weakly observed by the renderer, and the optional landmark term ($\gamma > 0$) adds geometric supervision. Each local basis $U_{a,j}$ is then obtained by PCA of the residuals of block j in the metric $M_{a,j}$ (Appendix A). Setting $\beta = \gamma = 0$ recovers the simple PCA of Section 3.2.1 within each attribute group and block. Evaluating with projected coefficients $c^\star_{a,j} = U_{a,j}^\top M_{a,j} r_{a,j}$ isolates the approximation quality of the basis from errors introduced by coefficient prediction.

Memory-budgeted component allocation. To maximize visual fidelity under a fixed memory budget, we select the numbers of components $K_{a,j}$ jointly across attribute groups and spatial blocks. Let $\lambda_{a,j,k}$ be the kth squared singular value in the PCA of block j (Appendix A). Since the PCA is computed in the metric $M_{a,j}$, $\lambda_{a,j,k}$ measures the variation of the residuals explained by component k weighted by its effect on the rendered image. Undoing the per-group normalization of $\widehat{H}_a$ and the weight β turns it into a value $v_{a,j,k}$ that is comparable across attribute groups, namely the estimated reduction of the squared image error of Eq. 5 summed over the samples (Appendix A). Given a basis memory budget $\mathcal{B}$ and b bytes per stored value, we maximize $\sum_{a,j} \sum_{k=1}^{K_{a,j}} v_{a,j,k}$ subject to $b \sum_{a,j} n_j d_a K_{a,j} \le \mathcal{B}$, and solve this allocation greedily by adding components in the order of their value per byte, $v_{a,j,k}/(b n_j d_a)$, until the budget is exhausted. This allocates capacity according to estimated visual benefit and storage cost without manually specifying ranks for individual attributes or regions, uses the precomputed spectra and requires no additional rendering.

3.2.3 REGRESSING BLENDSHAPE COEFFICIENTS

Projecting a residual of the host model onto the extracted basis provides accurate blendshape coefficients but requires evaluating the original animation network. To avoid this computation at inference time, we train a lightweight predictor

$$\widehat{c} = f_\phi(w, \theta) \in \mathbb{R}^K, \qquad K = \sum_{a,j} K_{a,j}, \tag{7}$$

which maps animation parameters θ and an identity descriptor w to coefficients of the fixed basis. We optimize parameters ϕ of a shallow MLP network f_ϕ, which we refer to as the *coefficient network*, by minimizing the distillation loss $\mathcal{L}_{\text{dist}} = \|\widehat{c} - c^\star\|_2^2$ on the same samples (w_i, θ_i) we use for the basis construction. Appendix E details the architecture and training of the coefficient networks. Identity conditioning allows the same expression or pose to produce subject-specific deformations despite sharing the basis across subjects.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Host models. We apply our method to three host models with distinct architectures. AGORA (Fazylov et al., 2025) generates heads with a GAN and animates canonical Gaussians with a separate convolutional deformation branch conditioned on FLAME expression and jaw parameters. FlexAvatar (Kirschstein et al., 2026) reconstructs a head from a single image and regenerates the attributes of all Gaussians at every frame with a transformer decoder conditioned on a FLAME-derived expression code. DynaAvatar (Kwon et al., 2026) reconstructs a full-body avatar with clothing dynamics from a single image and regenerates the Gaussians at every frame with a dynamic transformer conditioned on the current and 15 preceding SMPL-X poses. See Appendix B for more details.

Implementation details. The coefficient network of every host model is a shallow MLP, and for DynaAvatar it also depends on the window of preceding SMPL-X poses that the host model reads. The evaluation protocol of FlexAvatar fits the identity representation w to the input images of every test subject, and our model follows it with a per-subject *enrollment* that fine-tunes the coefficient network on outputs of the host model. We measure speed as the CPU time of one animation step, *i.e.* the computation of the Gaussian attributes $\hat{G}$ from the animation parameters θ without rendering, on the same desktop CPU for all methods. Appendix D and Table 5 give the settings of every host model.

Training data. Our method generates training data by running host models on samples of their inputs (w_i, θ_i). We take these samples from a subset of the training data of the corresponding host models and use them both to construct the basis and to train the coefficient network. For FlexAvatar we use 1,200 training identities with 8 expressions each, and for DynaAvatar we use frames sampled from its training sequences. AGORA is a generative model, hence we sample 10,000 identities from its latent space and combine each of them with 6 expressions of its training set FFHQ (Karras et al., 2019). The results in Tables 2–4 and Figure 4 are reported for identities excluded from this data.

4.2 RESULTS

Comparison to the state of the art. We evaluate every distilled model on the benchmark of its host model (Fazylov et al., 2025; Kwon et al., 2026; Kirschstein et al., 2026) and denote a host model distilled by our method with +GALA. We adopt metrics from previous publications and measure image quality (FID, PSNR, SSIM, LPIPS), the accuracy of expressions, poses and keypoints (AED, AED-jaw, APD, AKD) and the preservation of identity (ID, CSIM). Appendix F defines the metrics and the evaluation protocols.

Table 2 shows that all three distilled models closely follow their host models and outperform the competing methods on most metrics. For AGORA, FID increases from 3.17 to 3.47, while the animation and identity metrics remain within 0.023 of the host model and are the best (AED-jaw, ID, APD) or second best (AED) among the other animatable models. For DynaAvatar, our model has a 0.64 dB lower PSNR than its host and outperforms all competing methods on all three quality metrics, *e.g.* by more than 1.9 dB in PSNR. For FlexAvatar, our model obtains the best results on 8 of the 12 quality metrics, including one tie. Only LAM is competitive on portrait animation, where

Table 2: Comparison to the state of the art on the benchmark of each host model. In every panel the first row is the host model and the last row is the same model distilled by our method (+GALA). The rows in between are the methods compared in the paper of each host model, with the numbers reported there. The AGORA row is taken from Fazylov et al. (2025). The FlexAvatar and DynaAvatar rows are measured by us on the authors' released checkpoints, following their evaluation protocols (Appendix F). **Best** and second-best results, excluding the host model. EG3D and GGHead are not animatable. FPS is measured on a desktop GPU with a precomputed neutral avatar and mFPS on a mobile phone in a web browser. *As reported in the model's paper.

(a) Face 3D GANs on FFHQ

	FID↓	AED↓	AED-jaw↓	ID↑	APD↓	FPS↑	mFPS↑
AGORA	3.17	0.682	0.021	0.75	0.025	250*	1
EG3D	3.28	–	–	–	–	–	–
GGHEAD	4.06	–	–	–	–	–	–
Next3D	**3.18**	0.930	0.046	0.74	0.031	15*	–
GAIA	3.85	**0.530**	0.040	0.72	0.027	52*	–
AGORA + GALA (ours)	3.47	0.705	**0.022**	**0.76**	**0.026**	**1,275**	**60**

(b) Clothed full-body feed-forward on 4D-Dress

	PSNR↑	SSIM↑	LPIPS↓	FPS↑	mFPS↑
DynaAvatar	23.86	0.959	0.063	0.3	–
IDOL	21.31	0.948	0.077	–	–
PERSONA	19.46	0.943	0.098	–	–
LHM	21.03	0.950	0.085	–	–
DynaAvatar + GALA (ours)	**23.22**	**0.957**	**0.066**	**1,284**	**60**

(c) One-shot face feed-forward

	Speed		Portrait animation (VFHQ)						3D avatar creation (Ava256)					
			Self reenactment			Cross reenactment			Single-shot			Few-shot		
	FPS↑	mFPS↑	PSNR↑	LPIPS↓	AED↓	CSIM↑	AED↓	APD↓	PSNR↑	AKD↓	CSIM↑	PSNR↑	AKD↓	CSIM↑
FlexAvatar	140.7	–	23.29	0.099	0.075	0.652	0.226	0.025	19.28	7.48	0.508	21.50	6.17	0.755
Portrait4D-v2	–	–	21.34	0.144	0.117	0.656	0.268	0.273	11.9	7.77	**0.578**	–	–	–
GAGAvatar	–	–	21.83	0.122	0.111	0.633	0.253	0.247	12.7	**7.45**	0.555	–	–	–
LAM	280*	35*	**22.65**	**0.109**	0.102	0.651	0.250	0.356	13.1	11.2	0.411	–	–	–
GPAvatar	–	–	21.04	0.150	0.132	0.564	0.255	0.328	–	–	–	20.0	5.72	0.341
Avat3r	–	–	–	–	–	–	–	–	–	–	–	20.8	**5.66**	0.616
FlexAvatar + GALA (ours)	**1,296**	**60**	22.31	**0.109**	**0.091**	**0.680**	**0.245**	**0.029**	**19.25**	7.52	0.491	**21.31**	6.31	**0.740**

it has a 0.34 dB higher PSNR and the same LPIPS in self-reenactment. On Ava256 single-shot, our model stays within 0.03 dB PSNR of its host, which outperforms the compared one-shot methods by more than 3.8 dB PSNR under its published protocol (Kirschstein et al., 2026). The distilled model therefore retains this lead. With respect to its host, our model stays within 1 dB PSNR in self-reenactment and within 0.03 dB (single-shot) and 0.2 dB (few-shot) on Ava256.

At the same time, for the three tested host models our method reduces CPU time of their animation steps as 154→5.5ms, 234→4.4ms and 42.9s→6.1ms (16.1ms with skinning included) respectively (Table 5). This corresponds to speed-ups of about $\times 28$, $\times 54$ and $\times 7{,}000$ times, or $\times 2{,}700$ times for DynaAvatar when its skinning is included. With an animation step below 7ms on a desktop CPU for all three host models, our method enables real-time animation, even on a mobile phone, for host models that were not capable of it before. AGORA runs at 1 frame per second on the phone, and DynaAvatar, which models garment dynamics, reaches only 0.3 frames per second even on a desktop GPU, whereas our distilled models run at 60 frames per second on the phone (mFPS in Table 2). Our **supplementary video** demonstrates all three distilled models running in the web browser of a mobile phone (Appendix C).

Qualitative results. Figure 3 compares renderings of our method and of the host models for two examples per host model with pronounced facial expressions and body motion, and shows where the errors of our method are located. Our method faithfully reproduces both the facial expressions and the garment motion. With projected coefficients, it reproduces the host models with hardly visible errors in these examples. With predicted coefficients, which add the error of coefficient prediction, the remaining errors are small and localized at thin structures and at the outlines of the silhouettes, such as teeth and glasses for AGORA, mouth interior and hair boundary for FlexAvatar, and garment boundaries for DynaAvatar, and are mainly revealed by the error maps. Appendix J visualizes the local motions captured by individual blendshapes, and Appendix I presents more difficult examples, including a subject for which the coefficient network fails.

4.3 Ablation Study

Design choices of the basis construction. We evaluate the representation power of our basis, *i.e.* how well it reconstructs the output of the host model independently of coefficient prediction, by the

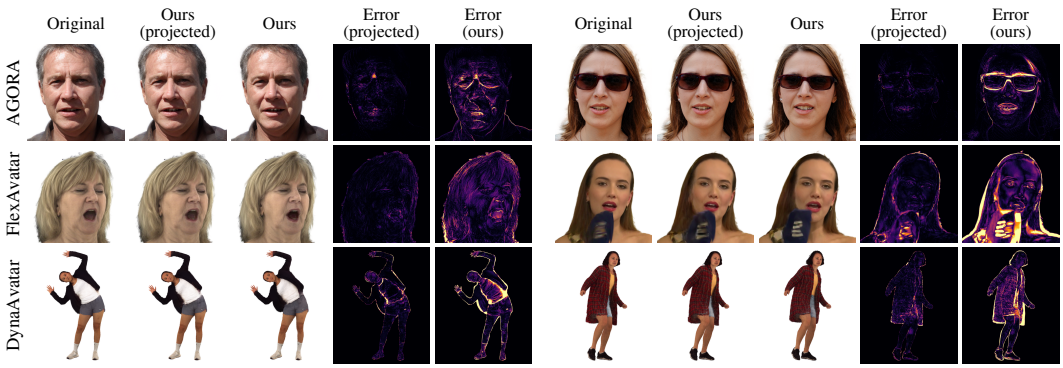

Figure 3: Qualitative comparison with the host models (*Original*), two examples per host model. *Ours (projected)* uses projected coefficients $c^\star$, which require running the host model and show what the basis can express (Section 3.2.3), and *Ours* the coefficients predicted by the coefficient network. Error maps show the absolute difference to the host model per pixel, averaged over the RGB channels (values in $[0, 1]$), from 0 (black) to 0.2 (yellow). Zoom in for details.

Table 3: Ablation of the basis construction. Row 1 of (a) is simple PCA (Section 3.2.1).

(a) Design choices of our method

Per-attribute bases	Rendering-aware PCA	Rendering-aware allocation	PSNR to host ↑ AGORA	Flex	Dyna
×	×	×	32.77	21.56	21.38
×	✓	✓	37.83	29.48	25.86
✓	×	✓	42.32	32.74	34.04
✓	✓	×	37.23	30.39	21.34
✓	✓	✓	**42.57**	**33.09**	**34.62**

(b) Number of blocks

B	PSNR to host ↑ AGORA	Flex	Dyna
1	34.35	26.19	26.04
4	37.00	28.80	28.63
8	38.04	30.13	30.64
16	40.46	31.54	32.65
32 (ours)	**42.57**	**33.09**	**34.62**

PSNR between the renderings of our model with projected coefficients $c^\star$ and of the host model on held-out identities, at the same memory budget for all compared variants. Table 3a compares our full method with simple PCA, the single global basis of Section 3.2.1 with the rank given by the budget, and with variants that revert one design choice while the others are kept. Simple PCA is 9.7, 11.5 and 13.2 dB below our full method for AGORA, FlexAvatar and DynaAvatar. Most of this gap comes from the two design choices that decide how the memory is spent: a joint basis over all attributes of a block costs 4.7, 3.6 and 8.8 dB, since a joint component stores all d channels instead of d_a and couples attributes with different scales, and scoring the components by explained variance instead of the value $v_{a,j,k}$ costs 5.3, 2.7 and 13.3 dB, since explained variance is not comparable between attribute groups with different units and spends most of the memory on attributes with little effect on the image. Rendering-aware PCA itself improves the directions of the basis by a further 0.25, 0.35 and 0.58 dB over Euclidean PCA ($\beta = \gamma = 0$) with the same numbers of components.

Allocation of components. Table 4 evaluates the need for allocating the numbers of components $K_{a,j}$ individually. We compare our greedy allocation (Section 3.2.2) with two simpler rules for the same bases and memory budget: one rank K shared by all attribute groups and blocks, and one rank K_a per attribute group shared by all blocks. Table 3a changes the values of the components, whereas here the values are fixed and only the constraints on the ranks change. A single rank reduces PSNR by 6.2, 1.2 and 8.5 dB, since it assigns the same number of components to every attribute group and to every block irrespective of their motion. Individual ranks per attribute group recover a large part of this gap. Our allocation per attribute group and block improves PSNR by a further 3.4, 0.1 and 0.7 dB. Given the values of the components, the greedy allocation requires no tuning and no rendering, and takes a few seconds.

Table 4: Allocation of components, PSNR to the host model.

Rank	AGORA	Flex	Dyna
Global, $K_{a,j} = K$	36.39	31.91	26.17
Per group, $K_{a,j} = K_a$	39.19	32.97	33.95
Per group and block, $K_{a,j}$ (ours)	**42.57**	**33.09**	**34.62**

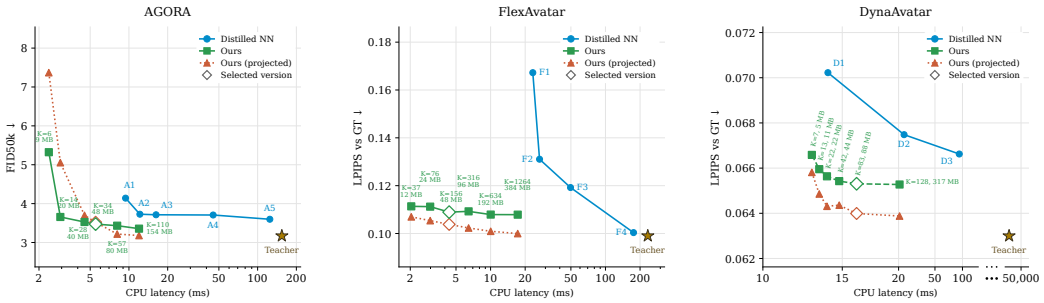

Figure 4: Comparison to alternative distillation: quality against the CPU time of the animation step, lower is better on both axes. *Teacher* is the host model, our models are labeled with the components per Gaussian and the basis size, and the diamond marks our final model.

Number of spatial blocks. Table 3b varies the number of spatial blocks B for a fixed basis size. Compared with a global basis per attribute group ($B = 1$), $B = 32$ local blocks improve PSNR by 8.2, 6.9 and 8.6 dB. A component of a local basis stores $n_j d_a$ instead of $n d_a$ values, hence more components fit into the same memory and they can be concentrated in the regions with complex motion. PSNR increases monotonically with B over the tested range, and we use $B = 32$ in all other experiments. For larger B we observed no consistent improvement with predicted coefficients, since more components make the coefficients harder to regress.

Comparison to alternative distillation. To further demonstrate the value of our linear blendshape distillation approach, we compare it to a generic method for neural network distillation. Knowledge distillation (Bucilă et al., 2006; Hinton et al., 2015) trains a smaller student network to reproduce the outputs of a frozen teacher on the inputs of the teacher. We distill students that are thinner and shallower copies of the animation network of corresponding host models. We train students of multiple sizes using the same training data as we use for training our method. Figure 4 compares this standard distillation with GALA for a range of memory budgets. For all three host models, none of the distilled networks approaches our method in terms of speed and accuracy. Additional details of this study are available in Appendix G.

5 Limitations and Future Work

While GALA already shows excellent results, its coefficient prediction leaves room for improvement. With projected coefficients $c^\star$ our models reproduce their host models closely, whereas with predicted coefficients their PSNR with respect to the host model is lower by 5.7 dB for AGORA and by 12.3 dB for FlexAvatar at a memory budget of 48 MB, and a larger budget mainly improves the results with projected coefficients (Figure 4). Another direction for improvement concerns the spatial partition, which is currently fixed to $B = 32$ blocks for all host models. Meanwhile an optimal partition may depend on the motion and could be selected jointly with the numbers of components $K_{a,j}$. Finally, our method inherits the limitations of host models, and our results suggest that avatar models could be trained directly with an identity-shared blendshape basis, which we leave for future work.

6 Conclusion

We introduced GALA, which replaces per-frame neural decoding of pretrained Gaussian avatar models with a shallow coefficient network and a linear combination of blendshapes. To the best of our knowledge, GALA is the first method to extract and leverage a blendshape basis shared across identities, whereas previous Gaussian blendshape methods require a separate basis for every subject. We apply GALA to three diverse avatar models for head and full-body animation and demonstrate excellent computational gains while closely preserving the quality of original models and outperforming recent baselines. GALA requires no retraining of original models, it accelerates CPU execution up to three orders of magnitude and enables animation to run at 60 fps on a mobile phone. These results demonstrate the effectiveness and broad applicability of proposed Gaussian blendshape distillation for efficient, high-fidelity on-device avatar animation.

AI USE STATEMENT

In this work we used generative AI tools to aid and polish the writing: proofreading, and finding and fixing typos, inconsistencies and incoherent passages. We have not used generative AI tools for the research ideation, the method, its implementation, the experiments or the interpretation of the results. We have reviewed all AI-assisted edits and take responsibility for the final content of this work.

REFERENCES

Florian Barthel, Wieland Morgenstern, Paul Hinzer, Anna Hilsmann, and Peter Eisert. CGS-GAN: 3D consistent Gaussian splatting GANs for high resolution human head synthesis. In *NeurIPS*, 2025.

C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Applied Numerical Mathematics*, 57(11-12):1214–1229, 2007.

Alexander W. Bergman, Petr Kellnhofer, Wang Yifan, Eric R. Chan, David B. Lindell, and Gordon Wetzstein. Generative neural articulated radiance fields. In *NeurIPS*, 2022.

Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *ACM SIGGRAPH Conf*, pp. 187–194, 1999.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *KDD*, pp. 535–541, 2006.

Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. FaceWarehouse: A 3D facial expression database for visual computing. *IEEE TVCG*, 20(3):413–425, 2014.

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, pp. 16102–16112, 2022.

Yufan Chen, Lizhen Wang, Qijing Li, Hongjiang Xiao, Shengping Zhang, Hongxun Yao, and Yebin Liu. MonoGaussianAvatar: Monocular Gaussian point-based head avatar. In *ACM SIGGRAPH Conf*, 2024.

Wei Cheng, Ruixiang Chen, Siming Fan, Wanqi Yin, Keyu Chen, Zhongang Cai, Jingbo Wang, Yang Gao, Zhengming Yu, Zhengyu Lin, Daxuan Ren, Lei Yang, Ziwei Liu, Chen Change Loy, Chen Qian, Wayne Wu, Dahua Lin, Bo Dai, and Kwan-Yee Lin. DNA-rendering: A diverse neural actor repository for high-fidelity human-centric rendering. In *ICCV*, pp. 19925–19936, 2023.

Xuangeng Chu and Tatsuya Harada. Generalizable and animatable Gaussian head avatar. In *NeurIPS*, 2024.

Xuangeng Chu, Yu Li, Ailing Zeng, Tianyu Yang, Lijian Lin, Yunfei Liu, and Tatsuya Harada. GPAvatar: Generalizable and precise head avatar from image(s). In *ICLR*, 2024.

Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. *IEEE TPAMI*, 23(6):681–685, 2001.

Yu Deng, Duomin Wang, Xiaohang Ren, Xingyu Chen, and Baoyuan Wang. Portrait4D: Learning one-shot 4D head avatar synthesis using synthetic data. In *CVPR*, pp. 7119–7130, 2024a.

Yu Deng, Duomin Wang, and Baoyuan Wang. Portrait4D-v2: Pseudo multi-view data creates better 4D head synthesizer. In *ECCV*, pp. 316–333, 2024b.

Helisa Dhamo, Yinyu Nie, Arthur Moreau, Jifei Song, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. HeadGaS: Real-time animatable head avatars via 3D Gaussian splatting. In *ECCV*, pp. 459–476, 2024.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Bernhard Egger, William A. P. Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhöfer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, Christian Theobalt, Volker Blanz, and Thomas Vetter. 3D morphable face models - past, present, and future. *ACM TOG*, 39(5):157:1–157:38, 2020.

Ramazan Fazylov, Sergey Zagoruyko, Aleksandr Parkin, Stamatis Lefkimmiatis, and Ivan Laptev. AGORA: Adversarial generation of real-time animatable 3D Gaussian head avatars. *arXiv preprint arXiv:2512.06438*, 2025.

Simon Giebenhain, Tobias Kirschstein, Martin Rünz, Lourdes Agapito, and Matthias Nießner. NPGA: Neural parametric Gaussian avatars. In *ACM SIGGRAPH Asia Conf*, pp. 127:1–127:11, 2024.

Yisheng He, Xiaodong Gu, Xiaodan Ye, Chao Xu, Zhengyi Zhao, Yuan Dong, Weihao Yuan, Zilong Dong, and Liefeng Bo. LAM: Large avatar model for one-shot animatable Gaussian head. In *ACM SIGGRAPH Conf*, pp. 27:1–27:13, 2025.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

M.F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, 18(3):1059–1076, 1989.

Sangeek Hyun and Jae-Pil Heo. GSGAN: Adversarial learning for hierarchical generation of 3D Gaussian splats. In *NeurIPS*, 2024.

Forrest Iandola, Stanislav Pidhorskyi, Igor Santesteban, Divam Gupta, Anuj Pahuja, Nemanja Bartolovic, Frank Yu, Emanuel Garbin, Tomas Simon, and Shunsuke Saito. SqueezeMe: Mobile-ready distillation of Gaussian full-body avatars. In *ACM SIGGRAPH Conf*, pp. 91:1–91:11, 2025.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pp. 4401–4410, 2019.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):139:1–139:14, 2023.

Tobias Kirschstein, Simon Giebenhain, Jiapeng Tang, Markos Georgopoulos, and Matthias Nießner. GGHead: Fast and generalizable 3D Gaussian heads. In *ACM SIGGRAPH Asia Conf*, pp. 126:1–126:11, 2024.

Tobias Kirschstein, Javier Romero, Artem Sevastopolsky, Matthias Nießner, and Shunsuke Saito. Avat3r: Large animatable Gaussian reconstruction model for high-fidelity 3D head avatars. In *ICCV*, pp. 12089–12100, 2025.

Tobias Kirschstein, Simon Giebenhain, and Matthias Nießner. FlexAvatar: Learning complete 3D head avatars with partial supervision. In *CVPR*, pp. 18193–18203, 2026.

Kevin Kwok. splat: WebGL 3D Gaussian Splat Viewer. `https://antimatter15.com/splat/`, 2023.

Joohyun Kwon, Geonhee Sim, and Gyeongsik Moon. Zero-shot reconstruction of animatable 3D avatars with cloth dynamics from a single image. In *CVPR*, 2026.

John P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Frédéric H. Pighin, and Zhigang Deng. Practice and theory of blendshape facial models. In *Eurographics (State of the Art Reports)*, pp. 199–218, 2014.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient ConvNets. In *ICLR*, 2017a.

Linzhou Li, Yumeng Li, Yanlin Weng, Youyi Zheng, and Kun Zhou. RGBAvatar: Reduced Gaussian blendshapes for online modeling of head avatars. In *CVPR*, pp. 10747–10757, 2025.

Tianye Li, Timo Bolkart, Michael J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM TOG*, 36(6):194:1–194:17, 2017b.

Shengjie Ma, Yanlin Weng, Tianjia Shao, and Kun Zhou. 3D Gaussian blendshapes for head avatar animation. In *ACM SIGGRAPH Conf*, 2024.

Julieta Martinez, Emily Kim, Javier Romero, Timur M. Bagautdinov, Shunsuke Saito, Shoou-I Yu, Stuart Anderson, Michael Zollhöfer, Te-Li Wang, Shaojie Bai, Chenghui Li, Shih-En Wei, Rohan Joshi, Wyatt Borsos, Tomas Simon, Jason M. Saragih, Paul Theodosis, Alexander Greene, Anjani Josyula, Silvio Maeta, Andrew Jewett, Simion Venshtain, Christopher Heilman, Yueh-Tung Chen, Sidi Fu, Mohamed Elshaer, Tingfang Du, Longhua Wu, Shen-Chi Chen, Kai Kang, Michael Wu, Youssef Emad, Steven Longay, Ashley Brewer, Hitesh Shah, James Booth, Taylor Koska, Kayla Haidle, Matthew Andromalos, Joanna Hsu, Thomas Dauer, Peter Selednik, Timothy Godisart, Scott Ardisson, Matthew Cipperly, Ben Humberston, Lon Farr, Bob Hansen, Peihong Guo, Dave Braun, Steven Krenn, He Wen, Lucas Evans, Natalia Fadeeva, Matthew Stewart, Gabriel Schwartz, Divam Gupta, Gyeongsik Moon, Kaiwen Guo, Yuan Dong, Yichen Xu, Takaaki Shiratori, Fabian Prada, Bernardo Pires, Bo Peng, Julia Buffalini, Autumn Trimble, Kevyn McPhail, Melissa Schoeller, and Yaser Sheikh. Codec avatar studio: Paired human captures for complete, driveable, and generalizable avatars. In *NeurIPS*, 2024.

Frederic I. Parke. Computer generated animation of faces. In *ACM Annual Conference*, pp. 451–457, 1972.

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, pp. 10975–10985, 2019.

Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. GaussianAvatars: Photorealistic head avatars with rigged 3D Gaussians. In *CVPR*, pp. 20299–20309, 2024.

Lingteng Qiu, Xiaodong Gu, Peihao Li, Qi Zuo, Weichao Shen, Junfei Zhang, Kejie Qiu, Weihao Yuan, Guanying Chen, Zilong Dong, and Liefeng Bo. LHM: Large animatable human reconstruction model for single image to 3D in seconds. In *ICCV*, pp. 14184–14194, 2025.

Geonhee Sim and Gyeongsik Moon. PERSONA: Personalized whole-body 3D avatar with pose-driven deformations from a single image. In *ICCV*, pp. 12670–12680, 2025.

L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519, 1987.

Jingxiang Sun, Xuan Wang, Lizhen Wang, Xiaoyu Li, Yong Zhang, Hongwen Zhang, and Yebin Liu. Next3D: Generative neural texture rasterization for 3D-aware head avatars. In *CVPR*, pp. 20991–21002, 2023.

Felix Taubner, Ruihang Zhang, Mathieu Tuli, Sherwin Bahmani, and David B. Lindell. MVP4D: Multi-view portrait video diffusion for animatable 4D avatars. In *ACM SIGGRAPH Asia Conf*, pp. 125:1–125:11, 2025a.

Felix Taubner, Ruihang Zhang, Mathieu Tuli, and David B. Lindell. CAP4D: Creating animatable 4D portrait avatars with morphable multi-view diffusion models. In *CVPR*, pp. 5318–5330, 2025b.

Phong Tran, Egor Zakharov, Long-Nhat Ho, Anh Tuan Tran, Liwen Hu, and Hao Li. VOODOO 3D: Volumetric portrait disentanglement for one-shot 3D head reenactment. In *CVPR*, pp. 10336–10348, 2024.

Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popovic. Face transfer with multilinear models. *ACM TOG*, 24(3):426–433, 2005.

Wenbo Wang, Hsuan-I Ho, Chen Guo, Boxiang Rong, Artur Grigorev, Jie Song, Juan Jose Zarate, and Otmar Hilliges. 4D-DRESS: A 4D dataset of real-world human clothing with semantic annotations. In *CVPR*, pp. 550–560, 2024.

Yue Wu, Yu Deng, Jiaolong Yang, Fangyun Wei, Qifeng Chen, and Xin Tong. AniFaceGAN: Animatable 3D-aware face image generation for video avatars. In *NeurIPS*, 2022.

Liangbin Xie, Xintao Wang, Honglun Zhang, Chao Dong, and Ying Shan. VFHQ: A high-quality dataset and benchmark for video face super-resolution. In *CVPRW*, pp. 656–665, 2022.

Hongyi Xu, Guoxian Song, Zihang Jiang, Jianfeng Zhang, Yichun Shi, Jing Liu, Wan-Chun Ma, Jiashi Feng, and Linjie Luo. OmniAvatar: Geometry-guided controllable 3D head synthesis. In *CVPR*, pp. 12814–12824, 2023.

Peizhi Yan, Rabab Ward, Qiang Tang, and Shan Du. Gaussian déjà-vu: Creating controllable 3D Gaussian head-avatars with enhanced generalization and personalization abilities. In *WACV*, pp. 276–286, 2025.

Zhengming Yu, Tianye Li, Jingxiang Sun, Omer Shapira, Seonwook Park, Michael Stengel, Matthew A. Chan, Xin Li, Wenping Wang, Koki Nagano, and Shalini De Mello. GAIA: Generative animatable interactive avatars with expression-conditioned Gaussians. In *ACM SIGGRAPH Conf*, pp. 143:1–143:10, 2025.

Youyi Zhan, He Wang, Tianjia Shao, and Kun Zhou. High-fidelity mobile avatars with pruned local blendshapes. In *CVPR*, pp. 32345–32356, 2026.

Yiyu Zhuang, Jiaxi Lv, Hao Wen, Qing Shuai, Ailing Zeng, Hao Zhu, Shifeng Chen, Yujiu Yang, Xun Cao, and Wei Liu. IDOL: Instant photorealistic 3D human creation from a single image. In *CVPR*, pp. 26308–26319, 2025.

Wojciech Zielonka, Timo Bolkart, and Justus Thies. Instant volumetric head avatars. In *CVPR*, pp. 4574–4584, 2023.

Wojciech Zielonka, Timo Bolkart, Thabo Beeler, and Justus Thies. Gaussian eigen models for human heads. In *CVPR*, pp. 15930–15940, 2025.

APPENDIX

Appendix A details the estimation of the rendering-aware metric and the construction of the basis in this metric space. Appendix B describes the three host models and Appendix C our mobile browser demos. Appendix D gives the implementation details of our method, Appendix E describes the coefficient networks, and Appendix F provides the evaluation details, including the evaluation of FlexAvatar on Ava256. Appendix G compares our method with distilled networks. Appendix H compares the distilled FlexAvatar and DynaAvatar models with per-subject blendshape avatars. Appendix I adds qualitative results, and Appendix J visualizes the motions of individual blendshapes. Please refer to the supplementary video for more.

A RENDERING-AWARE METRIC AND BASIS CONSTRUCTION

Estimation of the rendering Hessian. The rendering-aware metric of Section 3.2.2 is based on the Gauss–Newton Hessian $H = J^\top J$ of the rendered image with respect to the Gaussian attributes, which is too large to construct. We approximate it by its per-Gaussian diagonal blocks within each attribute group, which preserve the interactions among the channels of one Gaussian and one attribute group and discard the interactions between Gaussians and between attribute groups. We estimate these blocks with a stochastic estimator in the spirit of the trace and diagonal estimators of Hutchinson (1989) and Bekas et al. (2007). For a random vector ϵ in image space with entries ± 1 of equal probability, one backward pass through the renderer gives

$$g = J^\top \epsilon, \qquad \mathbb{E}[gg^\top] = J^\top \mathbb{E}[\epsilon\epsilon^\top] J = J^\top J. \tag{8}$$

For every Gaussian and attribute group a we therefore accumulate the $d_a \times d_a$ block of $gg^\top$ that belongs to the d_a channels of this Gaussian, and average over identities, animation parameters and cameras of the training samples, with 16 probes per frame. The estimate is symmetrized and normalized to unit mean trace per channel within each attribute group, which gives $\widehat{H}_a$. The normalization constant $\bar{h}_a$, the mean trace per channel before normalization, is used again by the allocation of components. The landmark matrix $\widehat{H}_a^{\mathrm{lmk}}$ is estimated in the same way, with the Jacobian of the landmark positions in place of J.

PCA in the metric space. For attribute group a and block j, let $M_{a,j}$ be the restriction of M_a of Eq. 6 to the n_j Gaussians of the block, and let $R_{a,j} \in \mathbb{R}^{n_j d_a \times N}$ contain the corresponding residuals as columns. The local basis is the solution of

$$\min_{U_{a,j},\{c_{i,a,j}\}} \sum_{i=1}^{N} \|r_{i,a,j} - U_{a,j} c_{i,a,j}\|^2_{M_{a,j}}, \qquad \|z\|^2_M = z^\top M z, \qquad U_{a,j}^\top M_{a,j} U_{a,j} = I. \tag{9}$$

Substituting $y = M_{a,j}^{1/2} r$ turns the objective into the Euclidean objective of Eq. 3, hence the problem is solved by uncentered PCA of

$$Y_{a,j} = M_{a,j}^{1/2} R_{a,j}. \tag{10}$$

If $V_{a,j}$ contains the top $K_{a,j}$ left singular vectors of $Y_{a,j}$, the basis and the projected coefficients are

$$U_{a,j} = M_{a,j}^{-1/2} V_{a,j}, \qquad c^\star_{a,j} = U_{a,j}^\top M_{a,j} r_{a,j}, \tag{11}$$

so that $U_{a,j}^\top M_{a,j} U_{a,j} = I$. Since $M_{a,j}$ is block diagonal with one $d_a \times d_a$ block per Gaussian, $M_{a,j}^{1/2}$ and $M_{a,j}^{-1/2}$ are computed per Gaussian. Setting $\beta = \gamma = 0$ gives $M_{a,j} = \alpha I$ and reduces the construction to the simple PCA of Section 3.2.1 applied to the block.

Value of a component. The allocation of Section 3.2.2 scores component k of attribute group a and block j by

$$v_{a,j,k} = \frac{\bar{h}_a}{\beta} \lambda_{a,j,k}, \tag{12}$$

for $\beta > 0$, where $\lambda_{a,j,k}$ is the kth squared singular value of $Y_{a,j}$ and $\bar{h}_a$ is the mean trace per channel of $\widehat{H}_a$ before normalization. Since $\widehat{H}_a$ is normalized separately for every attribute group, the singular values of different groups are not directly comparable, and the factor $\bar{h}_a/\beta$ undoes the normalization and the weight β of the rendering term. The value then combines the estimated reduction of the squared image error with the weighted reductions of the squared attribute error and, if $\gamma > 0$, of the landmark error, summed over the samples. Since the values are non-increasing within each local basis, the greedy allocation by value per byte retains the components of every local basis in spectral order.

B HOST MODELS

We apply our method to three host models, spanning different architectures, training objectives, and animation mechanisms. Together, these state-of-the-art models cover dedicated deformation networks, decoders with integrated expression conditioning, and temporally conditioned full-body animation. Table 5 lists the numbers of Gaussians and the attribute groups of the host models together with the settings of our distilled models.

AGORA (Fazylov et al., 2025) is a Gaussian avatar GAN trained adversarially on unstructured images of human heads. It follows the dual-branch design of Next3D (Sun et al., 2023) and GAIA (Yu et al., 2025), using GGHead's UV-space Gaussian representation (Kirschstein et al., 2024). A mapping network produces an identity code w, and an identity branch generates canonical Gaussian attributes on a UV grid aligned with FLAME. A separate deformation branch, conditioned on w and FLAME expression and jaw parameters θ, predicts attribute residuals. The identity branch is evaluated once per subject, whereas the deformation branch is evaluated at every frame. Identities can be generated by latent sampling or obtained by image inversion.

FlexAvatar (Kirschstein et al., 2026) reconstructs an animatable head from a single image using an encoder–decoder trained with rendering losses on monocular and multi-view portrait videos. The encoder produces an identity representation in template UV space. A FLAME-derived expression code conditions the decoder through cross-attention. Each animation frame therefore requires evaluating the full decoder to regenerate the attributes of all Gaussians. This model tests whether distillation remains effective when animation is integrated into the decoder rather than implemented by a separate deformation branch.

DynaAvatar (Kwon et al., 2026) reconstructs an animatable full-body avatar with clothing dynamics from a single image. Image encoders and a static transformer produce identity features on the SMPL-X template once per subject. A dynamic transformer, conditioned on a window of the current and 15 preceding SMPL-X poses, regenerates the attributes of template-anchored Gaussians at every frame, which are then posed by linear blend skinning. This model tests our method on full-body motion with garment dynamics that depend on the pose history. Its neural animation is by far the most expensive of the three models (Table 5).

C MOBILE BROWSER DEMO

Our real-time demos run in a web browser on desktop and mobile devices (Figure 5). They are built on the open-source WebGL Gaussian splat viewer of Kwok (2023), whose depth sort and splat rasterizer we keep and whose static point cloud we replace by our animated avatar. The neutral avatar and the blendshape basis are loaded once as textures. At every frame, the browser reads the animation parameters of the driving sequence, evaluates the coefficient network as fragment-shader passes on the GPU, forms the Gaussian attributes as the neutral avatar plus the coefficient-weighted sum of the blendshapes in a second pass, and hands the result to the sort and the rasterizer of the viewer. No host network runs in the browser. The FlexAvatar and DynaAvatar demos can show several avatars at once. The AGORA and FlexAvatar demos use the same viewer and differ only in the animation parameters they read, the FLAME expression and jaw parameters and the tracked expression code respectively, and in the sizes of the basis and the network (Table 5). The **supplementary video** shows the demos in action.

D IMPLEMENTATION DETAILS

Basis construction. We use the same configuration of our method for all host models unless stated otherwise. The Gaussians are partitioned into $B = 32$ spatial blocks, and a separate basis is constructed for every attribute group and block (Section 3.2.2). The rendering-aware metric M_a (Eq. 6) uses $\alpha = 1$ and $\beta = 16$, with $\beta = 64$ for DynaAvatar, and is estimated with 16 random probes per frame (Appendix A). The landmark term is used only for the positions of FlexAvatar ($\gamma = 24$), which is itself trained with a landmark loss. The numbers of components $K_{a,j}$ are allocated under a memory budget $\mathcal{B}$ on the storage of the basis (Section 3.2.2), which is listed in Table 5. The basis is stored in half precision ($b = 2$) for the two head models and in single precision ($b = 4$) for DynaAvatar.

Coefficient network. The architecture and training of the coefficient network f_ϕ (Section 3.2.3) are given in Appendix E. Table 5 lists its size for every host model.

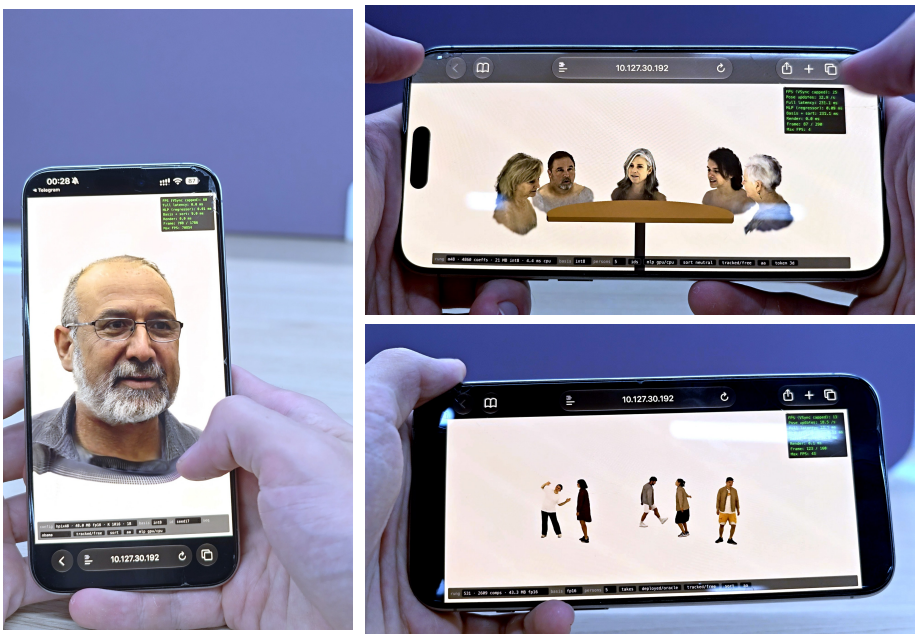

Figure 5: Our browser demos on a mobile phone, screenshots from the **supplementary video**. Left: AGORA + GALA driven by the tracked expressions of a video. Right: five FlexAvatar + GALA avatars, each reconstructed from one photo, around a table (top), and five DynaAvatar + GALA full-body avatars with clothing dynamics (bottom). The overlays show the frame rate and the timing of the animation step.

Table 5: Host models and the settings of our distilled models. Components are listed in total and per attribute group, in the order of the attribute groups. The animation step is timed on a CPU without rendering. For DynaAvatar, both the host model and ours additionally pose the Gaussians by linear blend skinning, which takes 10 ms per frame.

	AGORA	FlexAvatar	DynaAvatar
Gaussians n	228,693	58,361	40,000
attribute groups a	position, scale, rotation	position, scale, rotation, opacity, color	position, scale, rotation, opacity, color, skinning weights
animation parameters θ	FLAME expression and jaw	FLAME-derived expression code	current and 15 preceding SMPL-X poses
spatial blocks B	32	32	32
metric α, β, γ	1, 16, 0	1, 16, 24	1, 64, 0
bytes per value b	2	2	4
components K	1,016 (1,005 / 4 / 7)	4,860 (3,074 / 197 / 174 / 884 / 531)	2,686 (2,116 / 76 / 7 / 1 / 302 / 184)
memory budget $\mathcal{B}$	48.0 MB	48.0 MB	88.3 MB
coefficient network	1.6 MB	5.9 MB	60.7 MB
animation step, host	154 ms	234 ms	42,917 ms
animation step, ours	5.5 ms	4.4 ms	6.1 ms

Enrollment for FlexAvatar. The evaluation protocol of FlexAvatar fits the identity representation w to the input images of every test subject. Our model follows this protocol with a per-subject enrollment: given the fitted w and the expressions of the input images, we estimate a constant correction of the neutral avatar and fine-tune the coefficient network for 200 steps on outputs of the host model. The enrollment uses no target frames and no rendering.

Timing. We measure speed as the CPU time of one animation step, *i.e.* the computation of the Gaussian attributes $\hat{G}$ from the animation parameters θ without rendering. All methods are timed on

the same CPU, an Intel Core Ultra 9 285K, with eight threads and batch size one. For DynaAvatar, the linear blend skinning of 10 ms per frame is excluded from the animation step in Table 5 and included for all methods in Figure 4, whose time axis is cut before the host model at 43 s.

E COEFFICIENT NETWORKS

Table 6 summarizes the coefficient networks f_ϕ (Section 3.2.3) of the three distilled models of Table 2.

Table 6: Coefficient networks of the distilled models of Table 2. Hidden layers are listed with their widths. Parameters and multiply-accumulate operations (MAC) are those of one animation step of the network, without the basis. The FlexAvatar identity encoder (6.4 M parameters) runs once per identity and is not included.

	AGORA	FlexAvatar	DynaAvatar
identity descriptor w	latent of the neutral render (512)	embedding of the fitted avatar code (256)	identity tokens and shape parameters of the host encoder (3,082)
animation parameters θ	FLAME expression, jaw and eyelids (55)	articulation code (135)	16 SMPL-X frames $\times$ 79 values
hidden layers	256, 512	512, 512	per frame 512, 512, two temporal convolutions of 512, then 1,024, 1,024
activation	leaky ReLU	GELU	SiLU
outputs K	1,016	4,860	2,686
parameters	0.80 M	2.96 M	15.2 M
MAC per frame	0.80 M	2.96 M	42 M
training pairs (identities)	57,000 (9,500)	30,576 (2,184)	20,700 (1,035)
held out for selection	500 identities	98 identities	22 subjects
optimizer, learning rate	Adam, 10^{-3}	AdamW, $2 \cdot 10^{-3}$	AdamW, 10^{-3}
weight decay	0	10^{-6}	10^{-4}
batch size	256	all pairs	256
epochs run (selected)	1,008 (8)	512 (412)	60 (40)
gradient norm clipping	10	10	none
training time, one GPU	11 min	5 min	45 s
per-subject step	none	enrollment	none

Inputs and architecture. The identity descriptor w is computed once per identity. For AGORA it is the latent code of the neutral render. For FlexAvatar it is an embedding of the avatar code that the host model fits to the input image, produced by a small encoder with layers of 512 and 256 units that is trained jointly with the network. For DynaAvatar it consists of the pooled identity tokens and the shape parameters of the host's identity encoder. The animation parameters θ are the ones the host model reads: the FLAME expression, jaw and eyelid parameters for AGORA, the articulation code for FlexAvatar, and the window of the current and 15 preceding SMPL-X frames for DynaAvatar. The two head models use an MLP with two hidden layers on the concatenation of w and θ. The DynaAvatar network embeds every frame of the window with two layers, applies two temporal convolutions with kernel size 3 over the window, concatenates the flattened result with an embedding of w, and predicts the coefficients with two hidden layers. Its output is anchored at a reference pose, $\widehat{c} = f_\phi(w, \theta) - f_\phi(w, \theta_{\mathrm{ref}})$, so that the reference pose gives the neutral avatar for every subject. The last layer of every network outputs the K coefficients of all attribute groups and blocks.

Training. The targets are the projected coefficients $c^\star$ of the residuals of the host model with respect to the neutral avatar. The distillation loss of Section 3.2.3 is normalized per attribute group by the coefficient energy of that group on the training data. Every training pair combines an identity with animation parameters taken from elsewhere: an FFHQ expression for AGORA, an articulation code sampled from all training identities for FlexAvatar, and a motion window of another training subject for DynaAvatar, whose 1,035 identity instances are single source images of the 26 4D-Dress subjects and 136 DNA-Rendering (Cheng et al., 2023) actors in the training data of the host model.

All networks are trained in single precision with a cosine learning-rate schedule, and the checkpoint is selected on held-out identities, never on frames of the training identities.

F Evaluation Details

Benchmarks and metrics. Every distilled model is evaluated on the benchmark of its host model. AGORA is evaluated on FFHQ (Karras et al., 2019) with the protocol of Sun et al. (2023). FID compares the distributions of generated and real images and measures image quality, AED and AED-jaw measure the accuracy of the expression and jaw animation, ID measures the consistency of the identity under animation, and APD measures the accuracy of the head pose. DynaAvatar is evaluated on 4D-Dress (Wang et al., 2024), where PSNR, SSIM and LPIPS measure the pixel-wise, structural and perceptual similarity of the animated avatar to ground-truth images. FlexAvatar is evaluated on portrait animation on VFHQ (Xie et al., 2022) and on 3D avatar creation from one or a few images on Ava256 (Martinez et al., 2024). Here PSNR and LPIPS measure image quality with respect to the ground truth, AED, APD and the keypoint distance AKD measure animation accuracy, and CSIM, the cosine similarity of face recognition features, measures identity preservation. For AGORA, the rows of Table 2a are taken from Fazylov et al. (2025). For FlexAvatar and DynaAvatar, the host rows of Table 2b and c are measured by us on the checkpoints released by the authors, and the rows of the compared methods are quoted from the host papers.

Evaluation of DynaAvatar on 4D-Dress. We use the checkpoint released by Kwon et al. (2026) and follow their evaluation protocol on 83 sequences and 12,200 frames of the 4D-Dress test split. Our measurement of the host, 23.86 dB PSNR, 0.959 SSIM and 0.063 LPIPS, reproduces the published numbers of 23.74 dB, 0.960 and 0.064 within 0.12 dB. The host and our model are evaluated on the same renders.

Evaluation of FlexAvatar. On VFHQ we use the authors' main model FLEX-1 (`ckpt-900k`) and follow their released VFHQ-Test protocol. On Ava256 we use the released checkpoint `FLEX-2-avat3r-eval`, which according to the repository was not trained on the Avat3r test identities. The host row, our basis and our coefficient network are all computed from this checkpoint. Kirschstein et al. (2026) specify the structure of the Ava256 protocol, namely one frontal input frame with 10 expressions and 4 cameras per subject for the single-shot setting and four input images per subject following Avat3r for the few-shot setting, but not the particular subjects, sequences or expression frames, and the authors did not respond to our request for them. We therefore instantiate the protocol as follows. Kirschstein et al. (2026) evaluate on 5 subjects that are not identified, so we use all 8 unseen subjects of the authors' released Avat3r evaluation specification and the `EXP_free_face` sequence of each subject. The input is the subject's most frontal calibrated camera at the first frame of the sequence. The targets are evenly spaced frames of the sequence excluding the input frame, each rendered from the subject's 4 Avat3r input cameras. The host and our model are evaluated on exactly the same identities, input images and target frames. Our numbers for the host consequently differ from those in Kirschstein et al. (2026), *e.g.* 19.27 versus 16.9 dB single-shot PSNR over 8 and 5 subjects respectively, and the numbers of the compared methods in Table 2c are quoted from that paper. Since our method uses the host only through its outputs, the difference between the host and our model is expected to be unchanged under the original evaluation subset. The few-shot setting uses the authors' released evaluation specification for the Avat3r test identities (Kirschstein et al., 2025) with 8 subjects, 4 input views and 631 target images.

Ablations. Table 7 compares the simple PCA basis with our full method after coefficient prediction, on the benchmarks of Table 2. A coefficient network of the same architecture is trained for each basis. Our method is better on every quality metric except the identity metrics, by 2.4 dB PSNR for AGORA, 0.9 dB for FlexAvatar and 0.6 dB for DynaAvatar, and the higher ID and CSIM of simple PCA are a sign of under-animation. For the second rule of Table 4, the ranks K_a per attribute group are chosen without rendering: among the splits of the memory budget between the attribute groups, we take the one that maximizes the total value $v_{a,j,k}$ of the retained components, the same rendering-aware value that the greedy allocation uses.

G Comparison to Alternative Distillation

Distilled networks. The distilled networks of Figure 4 follow the standard recipe of knowledge distillation (Bucilă et al., 2006; Hinton et al., 2015): a student network is trained with a mean-squared loss on the outputs of the frozen host model, here normalized per attribute group, on the

Table 7: Simple PCA and our full method with predicted coefficients on the benchmarks of Table 2, at the memory budgets of Table 5. FlexAvatar is evaluated in self-reenactment on VFHQ, with the enrollment for both models. **Best** results. *With respect to the host model. †Cross-reenactment. The higher ID and CSIM of simple PCA reflect under-animation, since a static head obtains a CSIM of 0.813 on VFHQ.

	AGORA (FFHQ)					FlexAvatar (VFHQ)					DynaAvatar (4D-Dress)			
	PSNR* ↑	SSIM* ↑	FID↓	ID↑	AED↓	PSNR↑	SSIM↑	LPIPS↓	CSIM† ↑	AED† ↓	PSNR↑	SSIM↑	LPIPS↓	PSNR* ↑
Simple PCA	28.82	0.894	3.50	**0.797**	0.792	21.41	0.799	0.119	**0.75**	0.271	22.59	0.956	0.069	21.27
GALA (ours)	**31.18**	**0.929**	**3.47**	0.761	**0.705**	**22.31**	**0.818**	**0.109**	0.68	**0.245**	**23.22**	**0.957**	**0.066**	**23.87**

same samples as our bases and coefficient networks. Every student is a reduced copy of the animation network of its host model, and all other parts of the host model are kept frozen. For AGORA, the student is the convolutional deformation branch with fewer channels, initialized from scratch or, for A5, by magnitude pruning of the channels of the host (Li et al., 2017a). For FlexAvatar and DynaAvatar, the student keeps 1 to 8 of the blocks of the transformer that is evaluated at every frame, initialized from the host and reduced in width. The DynaAvatar students additionally attend over a farthest-point-sampled subset of the Gaussians and interpolate the predicted residual to all 40,000 Gaussians. Table 8 lists the size, the CPU time and the quality of every network.

Table 8: Distilled networks of Figure 4. For AGORA, channels are listed per resolution of the deformation branch. L and d are the number and the width of the transformer blocks, and N the number of attended Gaussians for DynaAvatar. Size is the storage of the student in single precision. CPU times are for the animation step at batch size one with eight threads on an Intel Core Ultra 9 285K, without the skinning of DynaAvatar. Quality is the vertical axis of Figure 4: FID on FFHQ for AGORA and LPIPS with respect to the ground truth for FlexAvatar and DynaAvatar.

	student	parameters	size	CPU	FID / LPIPS ↓
AGORA, deformation branch with 512 / 256 / 128 channels at 64^2 / 128^2 / 256^2					
A1	64 / 32 / 16 channels	0.56 M	2.2 MB	9.4 ms	4.14
A2	192 / 96 channels, no 256^2 block	1.03 M	4.1 MB	12.2 ms	3.73
A3	128 / 64 / 32 channels	0.89 M	3.6 MB	16.2 ms	3.71
A4	256 / 128 / 64 channels	1.77 M	7.1 MB	45.1 ms	3.71
A5	448 / 224 / 112 channels, pruned from the host	3.60 M	14.4 MB	124.4 ms	3.60
FlexAvatar, expression transformer with $L = 8$, $d = 768$					
F1	$L = 2$, $d = 192$, upsampler of the host at 3/8 width	4.6 M	18.5 MB	23.3 ms	0.167
F2	$L = 1$, $d = 576$	9.9 M	39.4 MB	26.7 ms	0.131
F3	$L = 2$, $d = 768$	27.2 M	108.7 MB	49.6 ms	0.119
F4	$L = 8$, $d = 768$	112.2 M	448.8 MB	175.5 ms	0.100
DynaAvatar, dynamic transformer with $L = 5$, $d = 1{,}024$, $N = 40{,}000$					
D1	$L = 1$, $d = 64$, $N = 252$	1.48 M	5.9 MB	3.9 ms	0.0702
D2	$L = 2$, $d = 320$, $N = 752$	11.6 M	46.3 MB	12.9 ms	0.0675
D3	$L = 2$, $d = 512$, $N = 2{,}500$	21.9 M	87.4 MB	82.2 ms	0.0666

Results. Figure 4 compares these networks with our method for a range of memory budgets. For all three host models, none of the distilled networks is both faster and more accurate than any of our models, and among the models that run in less than 80 ms the most accurate one is ours. For AGORA and DynaAvatar, none of the distilled networks reaches the quality of our model of Table 2. The best distilled AGORA network obtains an FID of 3.60 in 124 ms, whereas our model obtains 3.47 in 5.5 ms. For FlexAvatar, only the largest transformer approaches the LPIPS of the host model. It requires 175 ms, which is 40 times slower than our model. All distilled networks faster than 50 ms are less accurate than our smallest model, which has a 12 MB basis and takes 2 ms. The time of our method also grows slowly with memory, *e.g.* a 17 times larger AGORA basis requires 5 times more time, which we attribute mainly to the memory bandwidth of the CPU. Finally, the difference between our models with predicted and with projected coefficients shows the effect of coefficient prediction on the final quality. For larger budgets on FlexAvatar and DynaAvatar, projected coefficients bring our models close to the host model, which indicates that the remaining error is mostly due to coefficient prediction.

H Comparison to Per-Subject Blendshape Avatars

Per-subject methods fit an avatar to a video or a multi-view capture of each person, whereas our model initializes from one photo through the host model and reuses one shared basis.

Evaluation details. Table 9 compares FlexAvatar + GALA with RGBAvatar (Li et al., 2025) on 6 held-out identities of INSTA (Zielonka et al., 2023), and with GEM (Zielonka et al., 2025) and RGBAvatar on 3 identities of Ava256 (Martinez et al., 2024). RGBAvatar is fitted to the monocular video of each INSTA subject and, like GEM, to the 15-view capture of each Ava256 subject. Our model initializes from one photo of the subject. Self reenactment is scored on training frames of the per-subject fits (the first 250 frames of each INSTA video, and 120 Ava256 tuples per identity from 4 training cameras, none of them the camera of the input photo) and on held-out frames (the last 350 frames of each INSTA video, and 1,384 Ava256 tuples with unseen motion and mostly unseen views), cross reenactment on 3 VFHQ drivers without ground truth. Both per-subject fits are driven with their released pipelines: RGBAvatar with the drivers tracked by the same tracker and configuration as its training frames, and GEM with its image-based coefficient regressor relative to a neutral reference frame of each driver. Table 10 repeats the comparison with RGBAvatar on 3 short clips of the VFHQ test set (Xie et al., 2022), where RGBAvatar is fitted to the first 70% of each clip (127 frames on average) with the tracking of its released pipeline, FlexAvatar and our model use the first frame, and self reenactment is scored on the last 20% of each clip, with PSNR and LPIPS on 10 frames per clip. On 4D-Dress (Wang et al., 2024), WebAvatar (Zhan et al., 2026) is fitted to 24 views of each subject (999 frames per view, 4.8 h) and scored on a held-out take. Figures 6 to 8 show frames.

Results. In self reenactment, the per-subject fits score better: they have a lower AED on the held-out frames of every dataset, and RGBAvatar a higher PSNR on INSTA and VFHQ. This is expected. Each per-subject fit minimizes a photometric loss on the very sequence that self reenactment scores, from one video (INSTA, VFHQ) or from 15 views of the capture (Ava256), whereas our model never sees this sequence and is not optimized for its reconstruction. Our higher held-out PSNR on Ava256 comes from the camera of the input photo, on which the per-subject fits were not trained. The intended use of an avatar is cross reenactment, animation by the motion of another person, and there our model is ahead: it has a lower AED and APD than every per-subject fit on all three datasets and a higher CSIM on INSTA and VFHQ, whereas GEM keeps a higher CSIM on Ava256. Figures 6 to 8 show why: on their own held-out frames the per-subject fits are close to the ground truth, but under the motion of the driver RGBAvatar breaks up at head poses absent from its training video and GEM barely opens the mouth, whereas our model follows the driver. On 4D-Dress, WebAvatar obtains 22.78 dB, compared with 21.82 dB for DynaAvatar + GALA and 22.93 dB for DynaAvatar. On their own training frames, RGBAvatar reaches 33.21 dB on INSTA and 23.35 dB on Ava256, whereas GEM reaches 16.64 dB. Two properties of GEM explain its numbers. GEM does not model the neck and shoulders, which its pipeline masks out of its training images and which the ground truth keeps: this region covers 3% of the pixels but 45% of GEM's squared error, and GEM's own training image, scored as a render, reaches 19.3 dB. Its fit is also blurry from its first stage, which reaches 21.2 dB against its own neck-free training images before the basis and the coefficient regressor are fitted. On the head region alone, GEM reaches 20.05 dB on its training frames and RGBAvatar 23.92 dB.



Figure 6: Comparison to RGBAvatar on INSTA and Ava256. RGBAvatar is fitted to the video (INSTA) or to the 15-view capture (Ava256) of the subject, whereas the host model FlexAvatar and GALA use one photo. The self rows show held-out frames next to the ground truth, the cross rows are driven by VFHQ videos of other people (left) without ground truth.

Table 9: Comparison to per-subject blendshape avatars on 6 held-out identities of INSTA and 3 identities of Ava256. Input per subject: views, frames per view and time of the fit or of our enrollment, without face tracking. CSIM uses the input photo as reference. **Best** held-out and cross results, excluding the host model (†).

	Input per subject			Self reenactment					Cross reenactment		
				training frames		held-out frames					
	Views	Frames	Fit	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	AED↓	CSIM↑	AED↓	APD↓
INSTA											
RGBAvatar	1	2,790	55 s	33.21	0.0487	**30.23**	**0.0518**	**0.173**	0.524	0.898	0.0499
FlexAvatar†	1	1	∼1 min	24.58	0.0858	24.43	0.0675	0.311	0.612	0.700	0.0219
GALA (ours)	1	1	∼1 min	23.76	0.1152	23.80	0.0812	0.387	**0.639**	**0.800**	**0.0234**
Ava256											
GEM	15	2,921	27.8 h	16.64	0.2792	15.37	0.2949	**0.457**	**0.674**	1.006	0.0505
RGBAvatar	15	2,921	32.7 min	23.35	0.2076	18.12	0.2489	**0.457**	0.549	0.914	0.0385
FlexAvatar†	1	1	∼1 min	16.42	0.2974	19.22	0.2082	0.512	0.563	0.736	0.0245
GALA (ours)	1	1	∼1 min	16.49	0.2985	**18.81**	**0.2153**	0.579	0.590	**0.852**	**0.0247**

Table 10: Comparison to RGBAvatar on 3 short clips of the VFHQ test set. RGBAvatar is fitted to the first 70% of each clip, FlexAvatar and our model use the first frame, and self reenactment is scored on the last 20%. **Best** results, excluding the host model (†).

	Input per subject			Self reenactment, held-out frames			Cross reenactment		
	Views	Frames	Fit	PSNR↑	LPIPS↓	AED↓	CSIM↑	AED↓	APD↓
RGBAvatar	1	127	65 s	**20.70**	**0.1084**	**0.251**	0.451	0.860	0.0645
FlexAvatar†	1	1	∼1 min	19.72	0.1363	0.292	0.628	0.634	0.0188
GALA (ours)	1	1	∼1 min	19.15	0.1496	0.325	**0.666**	**0.787**	**0.0212**



Figure 7: Comparison to RGBAvatar on a VFHQ clip of Table 10. RGBAvatar is fitted to the first 70% of the clip, whereas the host model FlexAvatar and GALA use its first frame. Rows: a training frame of RGBAvatar, a held-out frame, and cross reenactment driven by a VFHQ video of another person (left).



Figure 8: Comparison to GEM on an Ava256 identity. GEM is fitted to the 15-view capture of the subject, whereas the host model FlexAvatar and GALA use one photo. Rows: a training frame of GEM from one of its training cameras, a held-out frame from the camera of the input photo, and cross reenactment driven by a VFHQ video of another person (left).

I ADDITIONAL QUALITATIVE RESULTS

Figure 9 shows more difficult examples in the layout of Figure 3, including a subject for which the coefficient network fails.

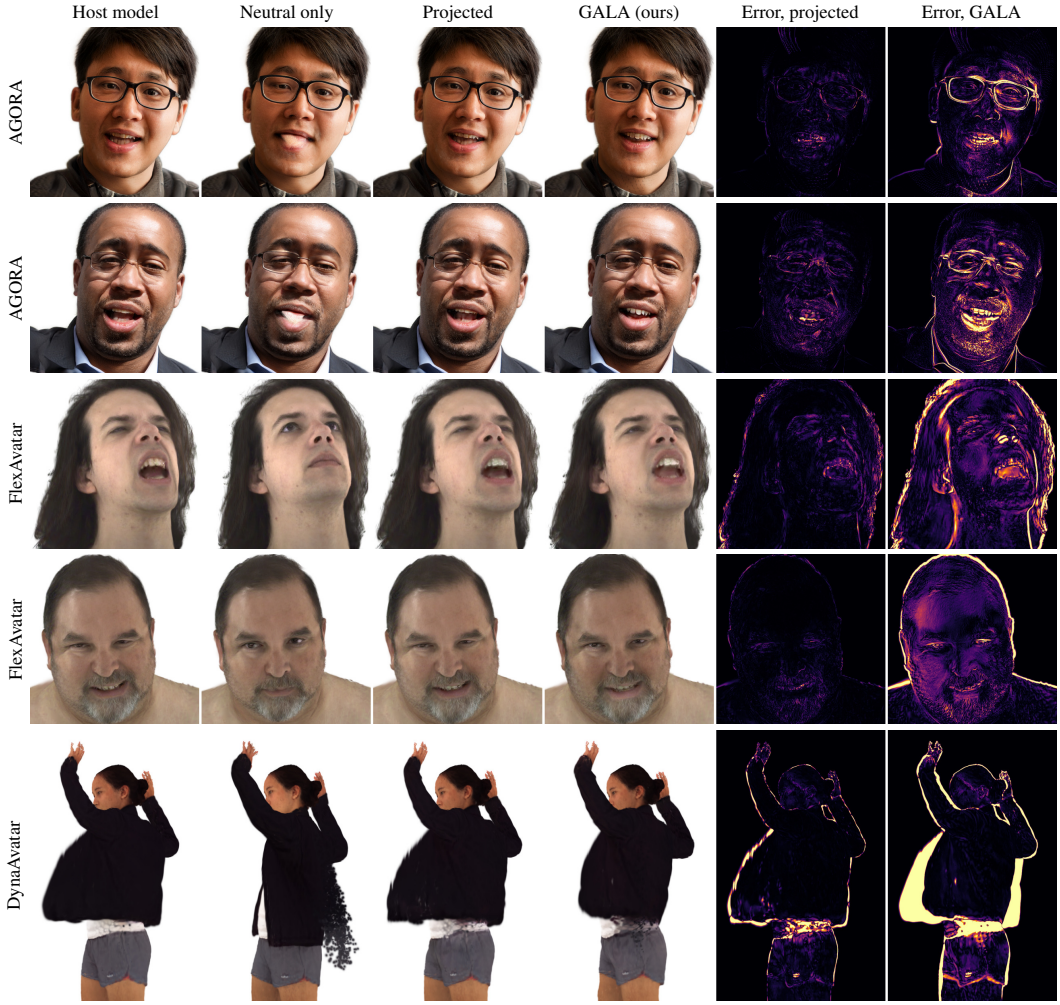

Figure 9: More difficult examples, with the error scale of Figure 3. *Neutral only* is the neutral avatar with all coefficients zero, *Projected* uses projected coefficients $c^\star$ and *GALA* the predicted ones. Rows 1 and 2: held-out AGORA identities at the frame with the largest residual of the host model. Rows 3 and 4: held-out Ava256 subjects in the few-shot and the single-shot setting. Row 5: the frame with the largest error of our model in the DynaAvatar sequence of Figure 3. In row 4 the coefficient network fails: 23.6 dB PSNR with respect to the host model with predicted coefficients, 40.2 dB with projected ones.

J VISUALIZATION OF BLENDSHAPES

Figure 10 visualizes motions captured by our basis. Each local basis $U_{a,j}$ is supported on one spatial block (Section 3.2.2), hence our blendshapes learn local motions that correspond to particular regions of the avatar, and the same blendshapes produce consistent motions for different identities. As examples we show blendshapes that control an eye for AGORA, the mouth for FlexAvatar, and a moving part of the garment for DynaAvatar, which is shown in the canonical A-pose.

Construction. For the head models, a Gaussian window is placed on a facial region of the shared template, and the components with at least a quarter of their displacement energy inside the window form a group (737 and 945 components for the AGORA rows, 349 and 343 for the FlexAvatar rows). The group is driven along the leading eigenvector of the covariance of its coefficients over

the training data, and the two states are the means of the coefficient vectors near the 2nd and the 98th percentile along this direction, applied through the window, with the amplitude for FlexAvatar limited to 14 mm for a head of 230 mm. For DynaAvatar, the first state is the neutral avatar, and the second sets one component to its 2nd or 98th percentile over the training data and the other coefficients to the mean of the nearest 10% of the coefficient vectors, attenuated toward the neutral avatar away from the component. The rows therefore show conditional activations of parts of the basis, not isolated basis vectors. The first column shows the window for the head models and the displacement field of the component for DynaAvatar, normalized per panel. The AGORA and FlexAvatar identities are held out, the DynaAvatar subjects belong to the training data. Bases: 154.1, 48 and 88.3 MB.

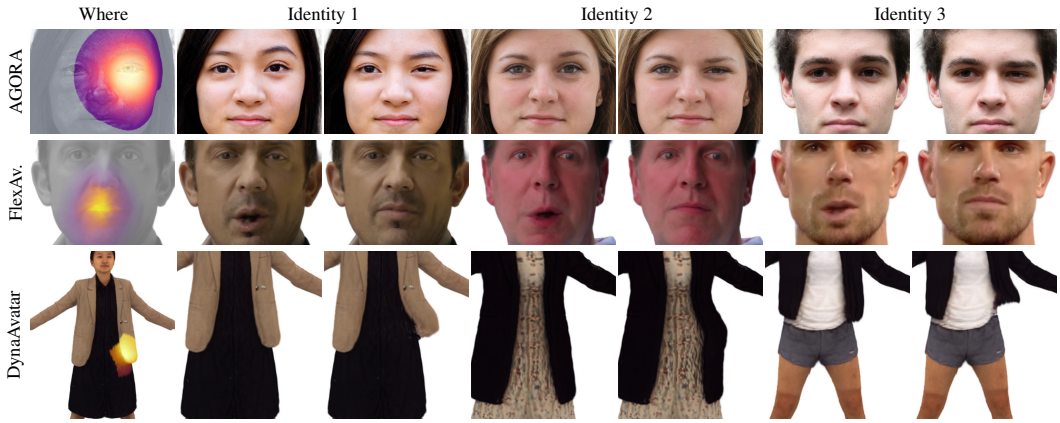

Figure 10: Local motions learned by our blendshapes. The first column highlights the region affected by the visualized blendshapes, and the other columns show two states of their motion for three identities: closing of an eye (AGORA), closing of the mouth (FlexAvatar) and lifting of the front part of an open garment (DynaAvatar).

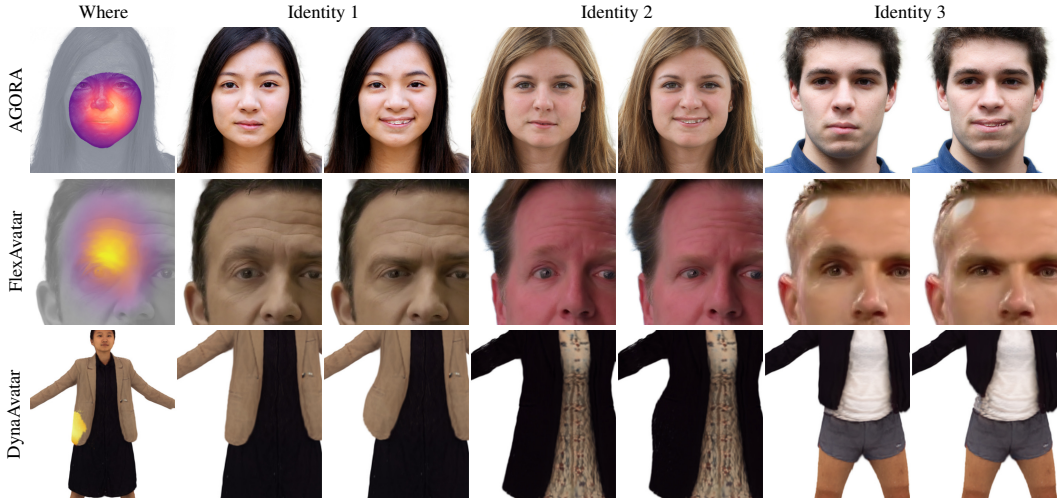

Figure 11: Further local motions, in the layout of Figure 10: a smile (AGORA), lowering of a brow together with the upper eyelid (FlexAvatar, cropped to the brow) and the opposite front part of the garment (DynaAvatar).