CharacterGAN: Few-Shot Keypoint Character Animation and Reposing

Tobias Hinz[1,2] Matthew Fisher[2] Oliver Wang[2] Eli Shechtman[2] Stefan Wermter[1]
[1]University of Hamburg [2]Adobe Research

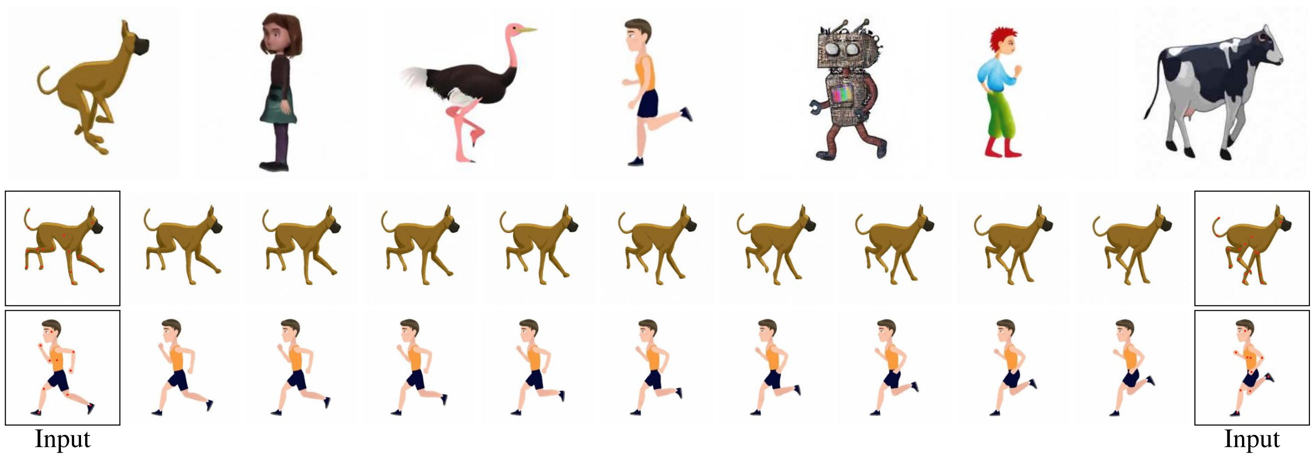

Figure 1: We train a generative model in a low-data setting (8 to 15 training samples) to repose and animate characters based on keypoint positions. The first row shows **video** results (indicated throughout the paper by 🎥), of our method driven by linearly interpolating input keypoints. Please view with Adobe Acrobat to see animations. The second and third rows show interpolated frames generated by our method between a *single* start and end frame (left and right columns).

Abstract

We introduce CharacterGAN, a generative model that can be trained on only a few samples (8 – 15) of a given character. Our model generates novel poses based on keypoint locations, which can be modified in real time while providing interactive feedback, allowing for intuitive reposing and animation. Since we only have very limited training samples, one of the key challenges lies in how to address (dis)occlusions, e.g. when a hand moves behind or in front of a body. To address this, we introduce a novel layering approach which explicitly splits the input keypoints into different layers which are processed independently. These layers represent different parts of the character and provide a strong implicit bias that helps to obtain realistic results even with strong (dis)occlusions. To combine the features of individual layers we use an adaptive scaling approach conditioned on all keypoints. Finally, we introduce a mask connectivity constraint to reduce distortion artifacts that occur with extreme out-of-distribution poses at test time. We show that our approach outperforms recent baselines and creates realistic animations for diverse characters. We also show that our model can handle discrete state changes, for example a profile facing left or right, that the different layers do indeed learn features specific for the respective keypoints in those layers, and that our model scales to larger datasets when more data is available.

1. Introduction

Modifying and animating artistic characters is a task that often requires experts to manually create many instances of the same character in different poses, which is a time-consuming and expensive process. Using video frame interpolation methods can reduce the overhead, but these methods do not leverage character-specific priors and so can only be used for small motions. Similarly, warping input frames directly with 2D handles cannot account for disocclusions or appearance changes between poses. In this work we have two main goals: (1) to generate high quality frames of an animated character based on a small number of examples, and (2) to generate these images based on a sparse set of keypoints that can be easily modified in real time.

To address both issues, we propose to train a conditional Generative Adversarial Network [11] (GAN) architecture that allows us to create new images of a character based on a set of given keypoint locations. We show that, with the right form of implicit biases, such a model can be trained in a few-shot setting (i.e., 10s of training images). In character animation, each training image has to be created manually, making it expensive to acquire the number of images required for training most conditional GAN models [16], and unlike recent work on single image generative models [33], we desire precise control over the generated image.

Our method consists of GAN that is trained on 8 – 15 images of a given character and its associated keypoints in different poses. One of the key challenges is that it is difficult for the model to learn which parts of a character should be occluded by other parts. E.g. when the hand of a humanoid character moves in front of the torso, either the hand should be visible at all times (if it is in the foreground) or only the torso should be visible if the hand moves “behind” the torso. To learn this ordering in a purely data-driven approach we need many images which we do not have in our setting. Instead, we propose to use user-specified *layers* for our keypoints, i.e. each keypoint lies in a given layer and we can introduce an ordering over those layers. For example, a humanoid character could be described by three layers, consisting of 1) the arm and leg in the “back” (i.e. occluded by the torso and other arm and leg), 2) the torso and head which are occluded by one arm and one leg and occlude one arm and one leg, and 3) the other arm and leg which occlude every other part of the character. Our model processes each of these layers independently, i.e. generates features for each layer without knowing the location of the keypoints in the other layers. We then use an adaptive scaling approach, conditioned on all keypoints, to spatially scale the features of each layer, before concatenating and using them to generate the final image.

Additionally, we can train our generator to predict the mask for a generated character which we found to be a robust way to identify and automatically fix unrealistic keypoint layouts at test time. Our model naturally learns to associate the keypoints with roughly semantically meaningful body parts for each layer, and can handle discrete state points that arise as a function of keypoint locations (e.g., switching between a profile facing left or right). Finally, to improve image quality, we use a patch-based refinement step [4] on the generated images based on [17].

We show via a number of qualitative and quantitative experiments that our resulting model allows for real-time reposing and animation of diverse characters and that our layering approach outperforms the more traditional conditioning approach of using all keypoints at once. Since we assume no prior knowledge about the modeled character our model can be applied to any shape and does not require additional data such as a 3D model or a character mesh. This allows our model to be applied in domains for which only limited data is available (e.g. artistic drawings or sprite sheets) without the need for additional manual input besides the keypoint labels. In summary, we introduce the following main contributions:

- We show that it is possible to train a GAN on only few images (8 – 15) of a given character to allow for few-shot character reposing and animation. By only conditioning the training on keypoints (instead of e.g. semantic maps) the trained model allows for character reposing in real-time without expert knowledge.
- By using a layered approach that explicitly encodes the ordering of different keypoints our model is able to model occlusions with only very limited training data.
- We introduce a mask connectivity constraint, where a jointly predicted mask can be used at test time to automatically fix keypoint layouts for which the model produces unrealistic outputs.

2. Related Work

Conditional GANs take as input some form of label which makes it possible to control the output of the generator to varying degrees and has also been shown to help with the training process. The label input can come in several forms, such as class conditioning [25], semantic maps [16], keypoints [31], or bounding boxes [14, 15]. However, most conditional GANs are trained with large datasets and are applied to broader domains, whereas we are interested in animating a *specific* character. In the following, we first focus on approaches in how to leverage small amounts of data to train GANs and conclude with a section about how the chosen conditioning method directly affects how the model can be used at test time.

Few-Shot Learning with GANs One promising approach to few-shot learning with GANs is to use fine-tune GANs that are pre-trained on large datasets. This can be achieved by fine-tuning a given model [42], by only training parts of the pre-trained model [27, 32, 26, 22], or transferring knowledge between different - but related - domains [41, 45]. However, these methods still rely on a model that is pre-trained on a large dataset in a very similar domain. As opposed to this we do not fine-tune a pre-trained model on a small dataset but instead train our model from scratch on limited available data.

Recent approaches show that applying data augmentation techniques during training of GANs directly is useful, especially when the dataset is small [37, 20, 46, 47]. One of the main insights of these approaches is that it is not sufficient to only augment real images since this leads the discriminator to learn that these augmented images are part of the real data. We also make heavy use of data augmentation in our

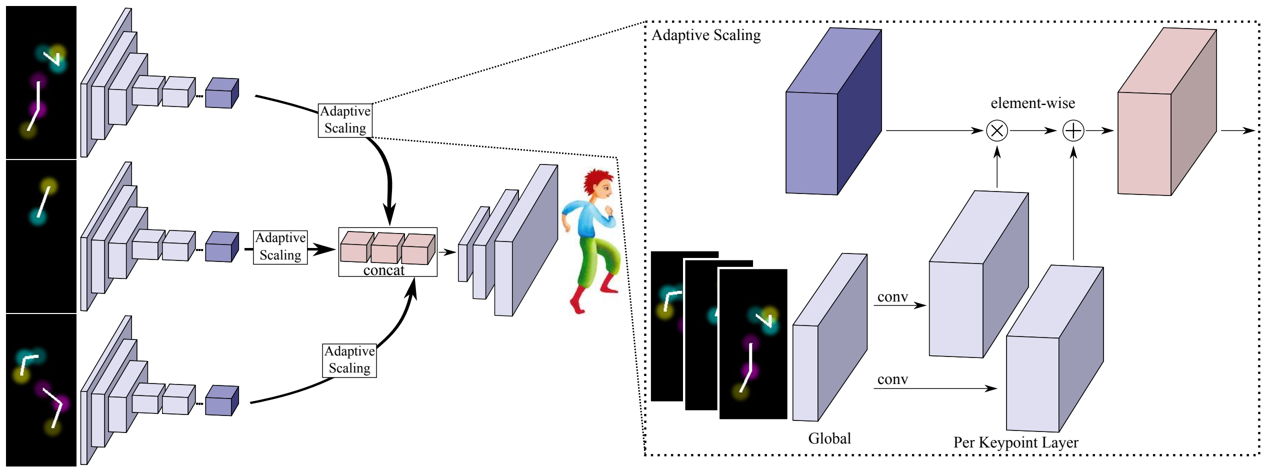

Figure 2: Our model processes keypoints which are split into individual layers. The resulting features for each keypoint layer are then scaled, concatenated, and used to generate the final image.

training process but use much less data (8 – 15 images) than the approaches mentioned here (usually 100+ images).

It is also possible to learn useful features from only a single image [49, 2] and that tasks such as texture synthesis [18, 21, 6, 48], image retargeting [34], inpainting and segmentation [38, 10], and unconditional image synthesis [33, 13] are possible with only a single image. Other approaches train GANs for image-to-image translation with only one pair of matching images [23, 5, 28, 39]. However, a model trained on just a single image without any other information can only generate limited variations of the training input. Therefore, we propose to increase the available information by slightly increasing the training set's size, which increases the model's capability to generate more variations of a learned object.

Editability An important characteristic of models for character reposing and animation is how the character is controlled. Existing approaches use driving videos [35], learn a distribution over poses for inbetweening [30], or map puppets to skeletons [8]. However, it is difficult to achieve a desired pose exactly with these approaches [35, 30] or requires expert knowledge to obtain the required training data [8]. Another approach is to condition the model on a semantic label map [7, 39] which can be changed at test time to reflect the new layout. However, modifying semantic maps or edge maps is not something that can be done on-the-fly, but instead takes time and skill. Finally, one could start with the character in a given pose and warp or stretch it until it reaches the desired pose [24] which might lead to unrealistic results if the end pose is too different from the starting pose. Given the previous limitations we only condition our model on high-level keypoints which are provided for a given character, making it easy and fast to generate new poses at test time by moving keypoints.

3. Methodology

In this section we describe our approach and how the model can be used at test time.

3.1. Input Requirements

In this work, we focus on character animation and reposing, where the images show the same character in different poses from the same viewpoint. Since our goal is to work in low data scenarios, we focus on cases with a small amount of images of the given character. Furthermore, the individual images can depict discrete appearance variations, e.g. different facial expressions. We do not make any other prior assumptions about the structure of the character. Our method requires an input image set, keypoints per image, keypoint connectivity information (essentially a skeleton), and a layer ordering for the keypoints. This information can be obtained easily as we only need labels for very few images. While the keypoints have to be labeled manually for each image, all other information is image independent and, thus, only needs to be defined once.

We experimented with various ways to input keypoints to the network, such as as individual channels, but found that representing keypoints as RGB Gaussian blobs performed the best. This also means that the input condition always has the same input dimensionality, independent of the number of keypoints, which is helpful as our model uses the same parameters to process these individual layers. Each keypoint is defined by a position $\{x, y\}$, three randomly chosen color values for the three RGB channels, and the values σ and r, where σ represents the falloff of the Gaussian distribution we use for blurring the keypoint, and r represent the radius of the final keypoint, which we define later. Discrete keypoints are modelled with individual RGB values, and in cases where different keypoints overlap we sum the respective colors.

3.2. Model

Our model consists of a generator and a discriminator (see Figure 2). The generator generates the image based on the keypoint locations while the discriminator is trained to distinguish between real and fake image-keypoint pairs. Finally, we apply a patch-match [4] based refinement step to improve the final quality of the generated images.

Generator and Discriminator We base our architecture on pix2pixHD [40]. However, while the discriminator is similar to the original pix2pixHD discriminator, we add several implicit biases to the generator reflecting our prior knowledge about the problem. Concretely, we know that the modeled characters are inherently three-dimensional, i.e. if some body parts are occluded by others they still exist even though they may not be visible. To address this, we split our characters into different layers, e.g. representing the "left" side of the character (e.g. left arm and leg), the "middle" part of the character (e.g. head and torso) and the "right" side of the character (e.g. right arm and leg). These layers can be modeled individually and can then be composed to form a final image. To model this, our generator processes each keypoint layer individually and learns a representation of each keypoint layer (see Figure 2).

Intuitively, we could set some features to zero if they are occluded by other features. For example, if the left hand is "behind" the torso for a given image, zeroing out the features for the left hand might make it easier for the generator to generate a realistic image. Conversely, if the right hand is "in front of" the torso, zeroing out the torso features at the location of the right hand might improve the performance. To address this, we incorporate an adaptive scaling technique [29] in which we scale the features of each layer before concatenating them. For this, we first learn an embedding of the keypoints and their layers ("Global" in *Adaptive Scaling* of Figure 2). Based on this embedding we then learn scaling parameters for each keypoint layer and use them to scale the features of each each layer. These scaled layer features are then concatenated and used to generate the final image.

Our discriminator takes the keypoint conditioning k concatenated with an RGB image as input and classifies it as either real or fake. We use two patch-discriminators [16], one of which operates on the full resolution image, while the second operates on the image down-scaled by a factor of two. We use a feature matching and an adversarial loss during training as defined by the pix2pixHD model [40]. The adversarial loss is the standard GAN loss L_{adv}:

$$\begin{aligned} \underset{G}{min}\ \underset{D}{max}\ \mathcal{L}_{\text{adv}} = \mathbb{E}_{(k,x)}[\log D(k,x)]+ \\ \mathbb{E}_{(k)}[\log(1-D(k,G(k))] \end{aligned} \tag{1}$$

where k is the keypoint condition and x is the corresponding real image. The feature matching loss L_{fm} stabilizes training by forcing the generator to produced realistic features at multiple scales and is defined as:

$$\underset{G}{min}\ \mathcal{L}_{\text{fm}} = \mathbb{E}_{(k,x)} \sum_{i}^{T} \frac{1}{N_i}[||D(k,x)-D(k,G(k))||_1] \tag{2}$$

where T is the number of layers in the discriminator and N_i is the number of elements in each layer. We add a perceptual loss [19, 44] to further improve the image quality. We use a VGG net to extract features from real and generated images and compute the perceptual loss L_{perc} as defined by [19]. Our final loss is the combination of these losses:

$$\underset{G}{min}\ \underset{D}{max}\ \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{fm}} + \mathcal{L}_{\text{perc}}. \tag{3}$$

Patch-based Refinement To further improve the final result we apply a patch-based refinement algorithm that replaces generated patches with their closest real patch. In our case, given a real and a generated image, for each patch in the generated image, we find the closest patch in the dataset of all real images using the Patch Match approximate nearest neighbor algorithm [4, 12], and replace the generated patches with their real equivalent [36]. We found that this approach often improves the sharpness and general image quality over the output of the generator (Figure 5).

Data Augmentation We employ both affine transformations and thin-plate-spline augmentation [39]. We use a mixture of horizontal and vertical translations and horizontal flipping and randomly sample a subset from these augmentation approaches at each training iteration. Thin-plate-spline (TPS) augmentation was introduced by Vinker et al. [39]. For this approach, the image is modeled as a grid and each grid point is then shifted by a random distance sampled from a uniform distribution. After this, a TPS is used to smooth the transformed grid into a more realistic warp. Using TPS augmentation results in warped images where parts of the image are stretched and elongated, adding further variation to the training data. All augmentations are applied to the given image and the associated keypoints and skeleton.

3.3. Editability

After our model is trained it offers a straightforward way to modify the pose of the character. Given an image of the character the user can drag keypoints to novel positions and the generator will generate the character in the new pose. We can also easily switch between different discrete states, and we provide two ways to do this. First, it can be handled completely automatic where the discrete state is determined solely by keypoint positions (e.g., for facing left vs right). Second, we provide the ability to have specific keypoints for individual states, such as smile vs frown, so a user can choose the desired expression at inference time. We also

Dataset	SinGAN [33]		ConSinGAN [13]		DeepSIM [39]		CharacterGAN	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
Watercolor Man	18.03±0.09	0.149±0.0032	21.65±0.32	0.102±0.0151	21.92±0.03	0.066±0.0002	**24.33±0.05**	**0.042±0.0002**
Watercolor Lady	18.03±0.09	0.159±0.0024	25.49±0.14	0.070±0.0012	26.21±0.04	0.048±0.0006	**28.27±0.03**	**0.037±0.0003**
Sprite Man	17.14±0.08	0.156±0.0063	23.19±0.17	0.087±0.0045	22.08±0.07	0.071±0.0001	**25.23±0.02**	**0.038±0.0006**
Dog	15.01±0.28	0.195±0.0078	19.24±0.27	0.125±0.0089	20.08±0.07	0.087±0.0010	**22.22±0.04**	**0.062±0.0006**
Ostrich	16.54±0.03	0.200±0.0023	23.30±0.18	0.103±0.0020	21.54±0.13	0.079±0.0006	**23.80±0.09**	**0.063±0.0005**
Cow	13.58±0.03	0.220±0.0016	18.71±0.12	0.133±0.0037	17.65±0.03	0.115±0.0013	**19.59±0.01**	**0.085±0.0005**

Table 1: Results of cross validation for the different models.

Dataset	CharacterGAN No Layer, No Scaling		CharacterGAN Layer, No Scaling	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
Watercolor Man	23.81 ± 0.15	0.049 ± 0.0050	24.29 ± 0.02	0.044 ± 0.0004
Watercolor Lady	28.23 ± 0.01	0.038 ± 0.0002	28.27 ± 0.05	0.037 ± 0.0003
Sprite Man	24.63 ± 0.12	0.041 ± 0.0020	25.12 ± 0.01	0.039 ± 0.0004
Dog	21.71 ± 0.10	0.068 ± 0.0007	22.14 ± 0.05	0.064 ± 0.0005
Ostrich	22.95 ± 0.04	0.068 ± 0.0007	22.97 ± 0.04	0.067 ± 0.0004
Cow	18.95 ± 0.08	0.094 ± 0.0021	19.52 ± 0.01	0.086 ± 0.0009

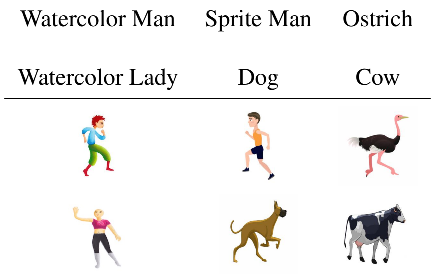

Table 2: Ablation study: results of cross validation for different parts of our model.

allow the user to optionally enable a mask-based connectivity correction. In this case, if the user positions keypoints too far from their input distribution, such that they could lead to unrealistic or undesired results, we can automatically modify the keypoint locations to achieve more realistic results.

Ensuring Mask Connectivity If the image background is of uniform color (e.g. white), or we have a segmentation network, we can automatically extract a foreground-background mask. We can use this mask as additional conditioning information during training, i.e. in this case the generator does not only generate the RGB image but also the mask, while the discriminator gets as input an image and its associated keypoints and mask.

At test time, if a keypoint is moved in a way that results in a layout that is too different from the ones seen at train time it can happen that the generator generates either "disconnected" body parts (e.g. the hand is not connected to the body) or introduces unwanted artifacts. Since the generator predicts the mask for the generated character we can use connected component analysis [9, 43] to check whether the generated mask is connected. We found two cases that often lead to a disconnected mask; if the disconnected part contains a keypoint the resulting character will be ripped apart or, alternatively, if the disconnected part does not contain a keypoint the generated image will often contain unwanted artifacts. In either case, we fix the result by automatically moving nearby keypoints in an iterative procedure until the predicted mask is fully connected again.

Given the last moved keypoint that caused the a disconnected region to appear, we identify the closest keypoint and move this keypoint in the same direction as the keypoint moved by the user by a fraction δ of the absolute distance. If $\delta = 1$ the keypoint is moved exactly in the same direction and by the same amount as the original keypoint and if $\delta = 0$ no keypoints are moved automatically at all. We found that a default of $\delta = 0.5$ achieved good results in a single iteration in most cases, however, this process can be repeated iteratively until the mask is connected (Figure 10).

4. Experiments

Baselines To compare our approach we adapt three generative baseline models from the single-image domain to our setting. SinGAN [33] and ConSinGAN [13] are trained on only a single image for tasks such as image generation and harmonization. Both models are trained in a multi-stage manner where the image resolution increases with each stage. While SinGAN trains each stage in isolation and freezes all previous stages, ConSinGAN trains the whole model end-to-end. We adapt both models by additionally conditioning the training at each stage on the image keypoints, i.e. each stage gets as input the keypoint condition in the respective image resolution. DeepSIM [39] trains a model specifically for image manipulation, where the conditioning input consists either of edge-maps, semantic labels, or a combination of both. The model is trained on only a single instance of an image and its conditioning information. We adapt the DeepSIM model to our setting by replacing the edge-map conditioning with keypoints, i.e., instead of the edge-map the input to the model is a map of keypoint locations.

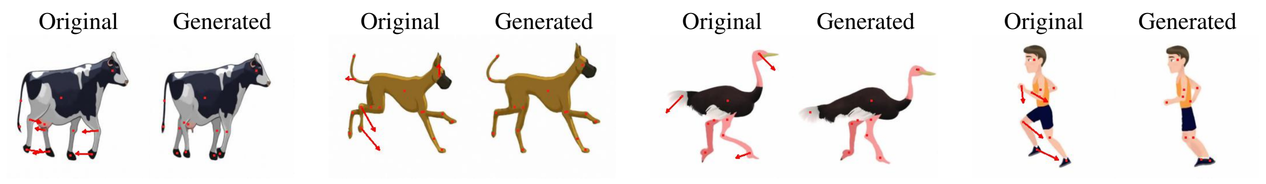

Figure 3: Examples from our model which was trained on only 8 – 12 images for each of the characters. Odd columns show the original image and our intended modifications, even columns show the output of our model.

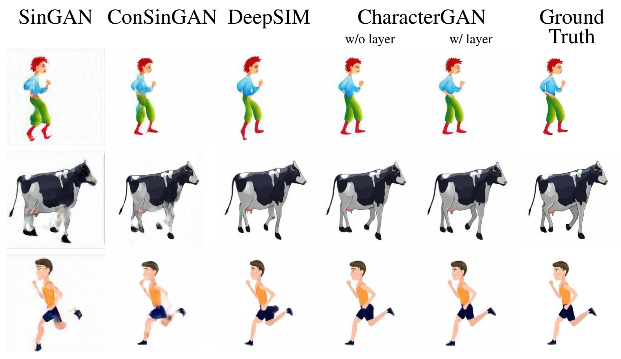

Figure 4: Qualitative examples of reconstructing held-out test images based on their keypoint locations.

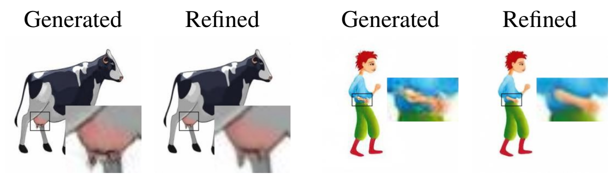

Figure 5: Effects of patch-based refinement (please view on screen and zoomed in).

Experiments We perform experiments on several different characters, including humans and animals. Our data comes from different sources such as drawings by artists [8] and characters taken from sprite sheets. For our characters we have 8 – 15 images which we manually label with keypoint locations, but we also show that our model is able to handle larger datasets. Our model itself can generate images in real-time and takes about 0.01 seconds for one image on an NVIDIA RTX 2080Ti (0.4 seconds on CPU). One iteration of evaluating the mask connectivity takes about 0.00002 seconds on CPU and the patch based refinement takes about 0.6 seconds (1.8 seconds on CPU).

4.1. Quantitative Evaluation

To the best of our knowledge there are no established quantitative evaluation metrics for few-shot character animation. [33] introduced the Single-Image FID (SIFID) score to evaluate single-image generative models. However, [13] report large variance in the SIFID scores and [32] report that models in the few-shot setting overfit to the FID metric. Other approaches use LPIPS [44] to compare a generated image with its ground truth counterpart. We use the Peak Signal-to-Noise Ratio (PSNR) and LPIPS as metrics to evaluate our model.

To evaluate our model we design an N-fold cross-validation for a given character with N images. Given a character we train our model N times on $N-1$ images were each image in the dataset is left out of training exactly once. At test time we generate the left-out image based on its keypoint layout and calculate the PSNR and LPIPS between the generated and ground-truth image. For each character we run the full N-fold cross-validation three times and report the average and standard deviation across the three runs. We perform all our quantitative evaluations without using the patch refinement step to evaluate the models directly. Some examples are shown in Figure 4.

Table 1 shows the results of our model compared to the baselines. Our model achieves the best LPIPS and PSNR for all characters. We observe that the PSNR is not always predictive of the (perceptive) quality of the generated image. In particular, SinGAN and ConSinGAN often generate images where the character exhibits disconnected body parts (e.g. the feet are not connected to the main body), but this is not represented in the PSNR, as feet and legs cover a relatively small area of the image.

Table 2 shows ablation studies with our model, where we train the same model without any layering, and with layering but no adaptive scaling. We can see that the layering approach improves the performance in all cases but for the "Watercolor Lady" character. This character is the only character of these that does not include occlusions, i.e. none of the bodyparts overlap each other which supports our hypothesis that the layering approach is mainly helpful for modeling occlusions in the low-data setting. Finally, adding the adaptive scaling further improves the performance, albeit not as much as the keypoint layering.

4.2. Qualitative Evaluation

Figure 3 shows visualizations of our model's capabilities on several different characters. All examples in this figure are trained on sprite sheets which contain 8 – 12 examples of the given character in different poses. Our model learns to generate realistic samples of four-legged animals, two-legged animals, and humanoid shapes. Furthermore, we see that our model can handle the movement of keypoints

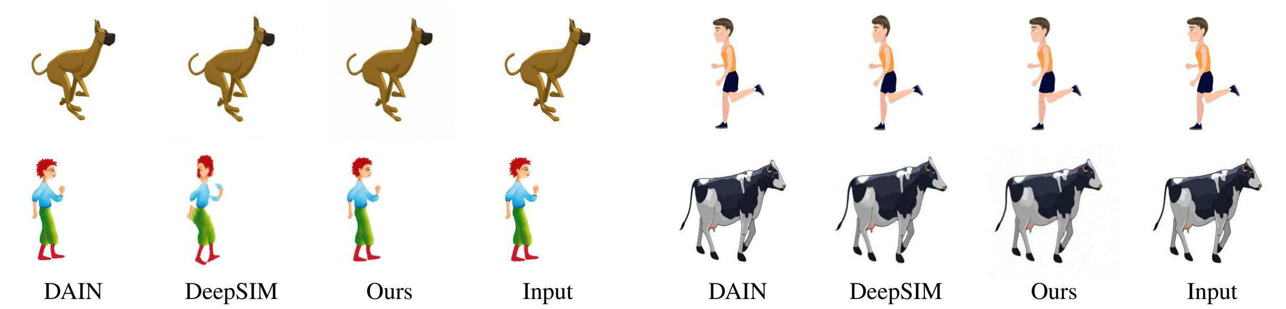

Figure 6: Comparison of our approach to frame interpolation [3] and DeepSIM [39] using the displayed input frames (🎥).

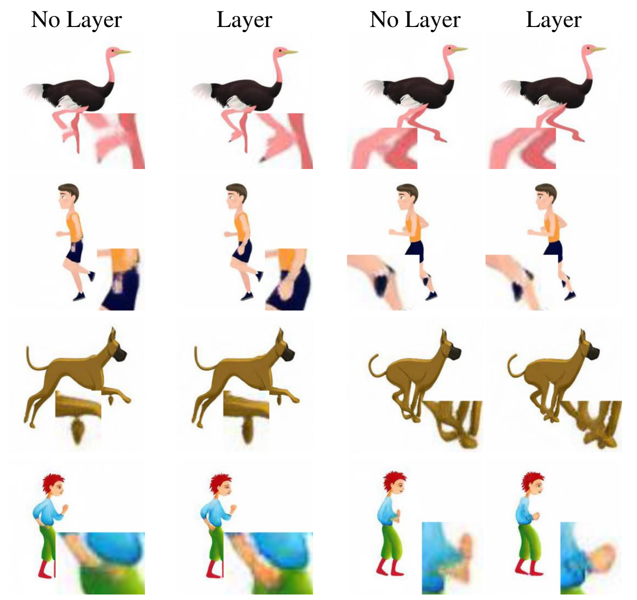

Figure 7: Comparison of layered vs non-layered at occlusions/overlaps.

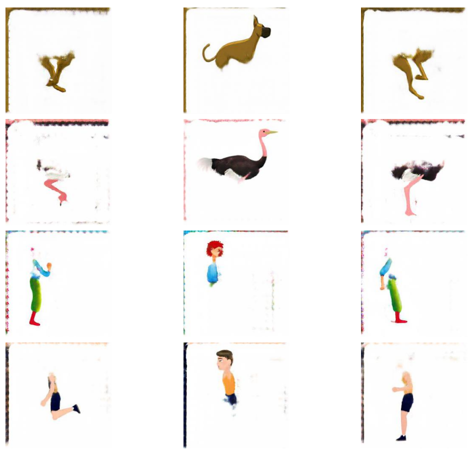

Figure 8: Visualization of what different layers learn (🎥).

that relate to relatively small character parts (e.g. individual feet) as well as keypoints that represent large body parts (e.g. head and torso). Even when we move multiple keypoints, the resulting image is realistic, adheres to the novel keypoint layout, and leaves areas of the character that were not modified unchanged. Figure 5 shows the effect of the patch-based refinement algorithm which often improves small details of the generated images.

Figure 1 and Figure 6 show how our model can be used for character animation. We use the poses from the training set as starting point and linearly interpolate the keypoints to generate the intermediate frames. Our model produces smooth and realistic results. We note that these intermediate keypoint locations could also be derived from other data e.g. by extracting keypoints from driving videos [1]. For comparison, we also show the results of a recent general purpose frame-interpolation model DAIN [3].

Mask-based Keypoint Refinement When moving certain keypoints to a new location we sometimes observe that this leads to "rips" in the generated character since the keypoint is too far away from the main body and such a configuration does not occur in training data. This can usually be fixed by moving the connected body part in the same direction as the original keypoint. Figure 10 shows examples of rips in the generated character (first two rows) and introduced artifacts (third row). The first column shows the original image and user specified modifications. The second and third columns show the predicted mask and generated image. The last column shows our model's final output after the respective connected keypoint was moved automatically. We can see that by enforcing the mask connectivity constraint we can get more realistic results in all cases.

Layered vs Non-layered Architecture Figure 7 shows qualitative comparisons between images generated by our architecture while using either our layered approach or a traditional, non-layered approach. As discussed previously, our layered approach is beneficial when several keypoints overlap in the 2D space, e.g. if a hand passes in front of a body or if two body parts are very close to each other. Figure 8 visualizes the features that our model learns for each keypoint layer. As we can see the model learns to only model the relevant keypoints and their associated body parts for each layer.

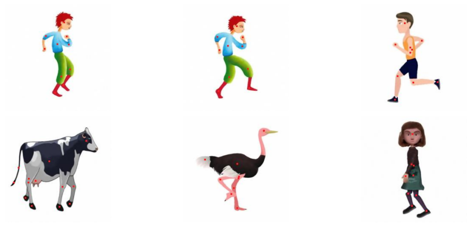

Figure 9: Discrete appearance changes based on keypoint location (🎥).

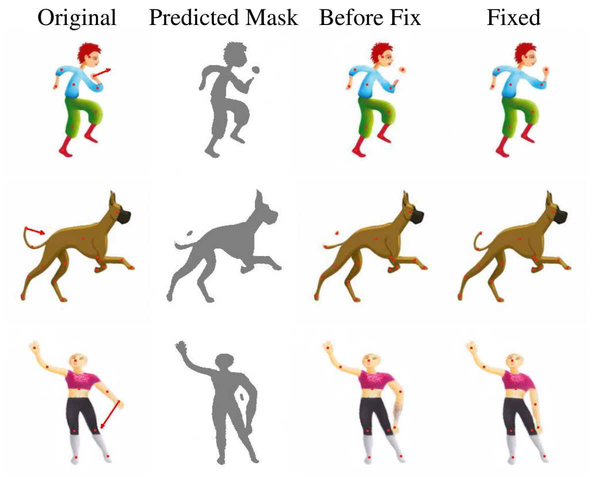

Figure 10: Enforcing mask connectivity at test time results in more realistic images.

Automatic Appearance Switching Our model does not only learn to associate discrete keypoints with given features, but also learns to associate different features with a keypoint based on its location relative to other keypoints. Figure 9 shows several examples in which we can see that individual parts get "flipped" as a function of the location of that keypoint with respect to the others. As in our previous examples, the features of unrelated keypoints are unaffected by this.

Appearance-specific Keypoints Figure 11 shows how our model is able to switch between discrete appearance states for given keypoints. Each of the characters shows different visual features during training which we encode as different keypoint conditions. At test time we can switch between these different states to combine novel poses with any of the discrete visual expressions. Note that, again, our model learns to associate good features with the given keypoints, allowing us to model the poses independently of the discrete keypoint states.

Scaling to Larger Datasets While we show that our model performs well with only 8 – 15 training images, we also evaluate our model on characters for which we have more training images (≥ 50). Figure 12 shows how our model scales with larger datasets. We see that more data is especially helpful when there are overlaps and occlusions.

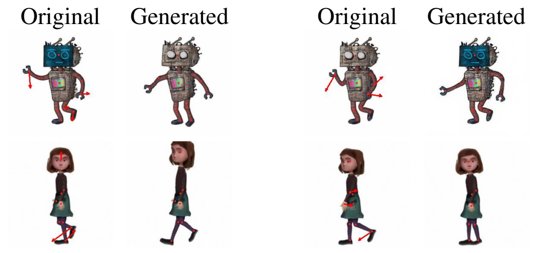

Figure 11: Discrete appearance change based on keypoint selection. In this example, the user not only moves keypoints to generate a new pose, but switches the IDs of keypoints to change expression, such as the color and rotation of the characters' faces.

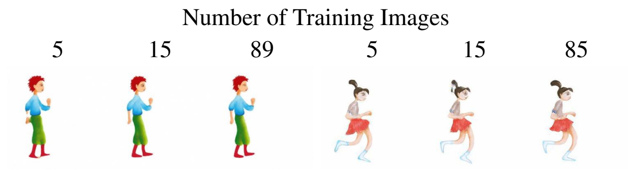

Figure 12: Here we show how performance improves with more training images (🎥).

While the models trained on only 5 or 15 images have difficulty modeling this (e.g. when the hand moves in front of the body), the models which are trained on more images perform much better.

Limitations Our model addresses the main challenge of correctly modeling (dis)occlusions based on limited training data. However, our model still has no explicit understanding about any underlying 3D representation of the character and Figure 12 shows how modeling occlusions gets worse with fewer training images. This could be addressed by future work, e.g. by adding known character priors into the model or by incorporating some form of 3D understanding.

5. Conclusion

In this work we show how to train GANs on few examples (8 – 15 images) of a given character for few-shot character reposing and animation. The model is easy to use, requires no expert knowledge, and our layering approach produces realistic results for novel poses and occlusions. We can perform character reposing in real-time through moving around keypoints and can animate character by interpolating keypoints. In the future, this can be used with other approaches, e.g. by extracting keypoints from driving videos for character animation. Through the use of a predicted foreground mask we can also automatically fix keypoint layouts that lead to unrealistic character poses. Finally, we show that our model learns discrete state changes based on keypoint locations, associates keypoints and their layers with semantic body parts, and scales to larger datasets.

Acknowledgements

The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169). We also thank Zuzana Studená who produced some of the artwork used in this work.

References

[1] Kfir Aberman, Jing Liao, Mingyi Shi, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Neural best-buddies: Sparse cross-domain correspondence. *ACM Transactions on Graphics (TOG)*, 37(4):69, 2018. 7

[2] Yuki M Asano, Christian Rupprecht, and Andrea Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. In *International Conference on Learning Representations*, 2020. 3

[3] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 3703–3712, 2019. 7

[4] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 28(3):24, 2009. 2, 4

[5] Sagie Benaim, Ron Mokady, Amit Bermano, Daniel Cohen-Or, and Lior Wolf. Structural-analogy from a single image pair. *arXiv preprint arXiv:2004.02222*, 2020. 3

[6] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial gan. In *International Conference on Machine Learning*, pages 469–477, 2017. 3

[7] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5933–5942, 2019. 3

[8] Marek Dvorožňák, Wilmot Li, Vladimir G. Kim, and Daniel Sýkora. ToonSynth: Example-based synthesis of hand-colored cartoon animations. *ACM Transactions on Graphics*, 37(4), 2018. 3, 6

[9] Christophe Fiorio and Jens Gustedt. Two linear time union-find strategies for image processing. *Theoretical Computer Science*, 154(2):165–181, 1996. 5

[10] Yossi Gandelsman, Assaf Shocher, and Michal Irani. "double-dip": Unsupervised image decomposition via coupled deep-image-priors. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, 2019. 3

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 2

[12] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001. 4

[13] Tobias Hinz, Matthew Fisher, Oliver Wang, and Stefan Wermter. Improved techniques for training single-image gans. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1300–1309, 2021. 3, 5, 6

[14] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Generating multiple objects at spatially distinct locations. *International Conference on Learning Representations*, 2019. 2

[15] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Semantic object accuracy for generative text-to-image synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 1125–1134, 2017. 2, 4

[17] Ondrej Jamriska. Ebsynth: Fast example-based image synthesis and style transfer. https://github.com/jamriska/ebsynth, 2018. 2

[18] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. In *Advances in Neural Information Processing Systems Workshop*, 2016. 3

[19] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016. 4

[20] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems*, 2020. 2

[21] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716, 2016. 3

[22] Yijun Li, Richard Zhang, Jingwan Cynthia Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[23] Jianxin Lin, Yingxue Pang, Yingce Xia, Zhibo Chen, and Jiebo Luo. Tuigan: Learning versatile image-to-image translation with two unpaired images. In *European Conference on Computer Vision*, 2020. 3

[24] Songrun Liu, Alec Jacobson, and Yotam Gingold. Skinning cubic bézier splines and catmull-clark subdivision surfaces. *ACM Transactions on Graphics (TOG)*, 33(6):1–9, 2014. 3

[25] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[26] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze discriminator: A simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*, 2020. 2

[27] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2750–2758, 2019. 2

[28] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, 2020. 3

[29] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 4

[30] Omid Poursaeed, Vladimir Kim, Eli Shechtman, Jun Saito, and Serge Belongie. Neural puppet: Generative layered cartoon characters. In *IEEE Winter Conference on Applications of Computer Vision*, pages 3346–3356, 2020. 3

[31] Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *Advances in Neural Information Processing Systems*, pages 217–225, 2016. 2

[32] Esther Robb, Wen-Sheng Chu, Abhishek Kumar, and Jia-Bin Huang. Few-shot adaptation of generative adversarial networks. *arXiv preprint arXiv:2010.11943*, 2020. 2, 6

[33] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4570–4580, 2019. 2, 3, 5, 6

[34] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and retargeting the dna of a natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4492–4501, 2019. 3

[35] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *Advances in Neural Information Processing Systems*, 2019. 3

[36] Ondřej Texler, David Futschik, Jakub Fišer, Michal Lukáč, Jingwan Lu, Eli Shechtman, and Daniel Sỳkora. Arbitrary style transfer using neurally-guided patch-based synthesis. *Computers & Graphics*, 87:62–71, 2020. 4

[37] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. Towards good practices for data augmentation in gan training. *arXiv preprint arXiv:2006.05338*, 2020. 2

[38] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 9446–9454, 2018. 3

[39] Yael Vinker, Eliahu Horwitz, Nir Zabari, and Yedid Hoshen. Deep single image manipulation. *arXiv preprint arXiv:2007.01289*, 2020. 3, 4, 5, 7

[40] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 4

[41] Yaxing Wang, Abel Gonzalez-Garcia, David Berga, Luis Herranz, Fahad Shahbaz Khan, and Joost van de Weijer. Minegan: effective knowledge transfer from gans to target domains with few images. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 9332–9341, 2020. 2

[42] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *European Conference on Computer Vision*, pages 218–234, 2018. 2

[43] Kesheng Wu, Ekow Otoo, and Arie Shoshani. Optimizing connected component labeling algorithms. In *Medical Imaging 2005: Image Processing*, volume 5747, pages 1965–1976, 2005. 5

[44] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 586–595, 2018. 4, 6

[45] Miaoyun Zhao, Yulai Cong, and Lawrence Carin. On leveraging pretrained gans for generation with limited data. In *International Conference on Machine Learning*, 2020. 2

[46] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. In *Advances in Neural Information Processing Systems*, 2020. 2

[47] Zhengli Zhao, Zizhao Zhang, Ting Chen, Sameer Singh, and Han Zhang. Image augmentations for gan training. *arXiv preprint arXiv:2006.02595*, 2020. 2

[48] Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018. 3

[49] Maria Zontak and Michal Irani. Internal statistics of a single natural image. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 977–984, 2011. 3