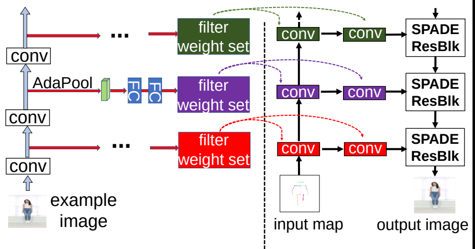
We thank all reviewers for their positive and constructive comments, such as the application important, the results impressive, and the method generalizable. Below we first address the common questions, and then questions by individual reviewers.



(a) weight generation scheme **(b) modified SPADE**

Fig A: Our network architecture

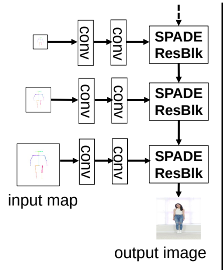


Fig B: Original SPADE

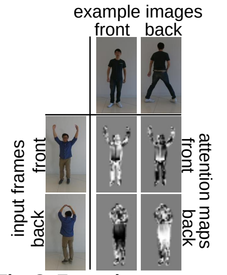


Fig. C: Example attn maps

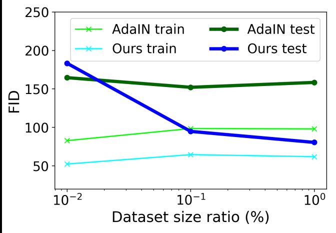


Fig. D: Comparison to AdaIN with varying dataset sizes.

Details of network architecture. We are sorry there were not enough details provided in the paper due to the page limit. In Fig. A, we show the network architecture when only one example image is used (i.e. without the attention mechanism). Compared to the original SPADE generator (Fig. B), we made two major modifications. First, instead of generating the affine parameters at each layer independently, we reuse features from the previous layer when generating parameters in the next layer (Fig. A(b)). Second, the weights in SPADE are determined on the fly (Fig. A(a)). After we encode the example image, we apply adaptive pooling (AdaPool) to make it fixed spatial size. The results are fed to two fully connected layers to generate the weights, which are used in corresponding convolutions (denoted by different colors).

	AdaIN (avg)	AdaIN (attn)	Ours (attn)
FID	129.90	113.83	***76.53***

Table A: Comparison to AdaIN *(5 example images).*

	Ours (1 exp)	Ours (50 exps)	Ours (50 exps + finetune)	vid2vid
FID	56.99	51.10	***43.04***	47.19

Table B. Comparison to vid2vid

Attention mechanism. We combine features from multiple example images before feeding them to the AdaPool layer in Fig. A(a). The attention maps are soft spatial maps of size $\mathbb{R}^{K\times N\times N}$, where K is the number of examples and N is the spatial dimension. The map determines that for each spatial location in the output (x, y), which spatial location in which example image (k, x_k, y_k) carries most relevant information. For example, when example images include both front and back of the target person, the attention maps can help capture corresponding body parts during synthesis (Fig. C).

Comparison to AdaIN. In AdaIN, information from the example image is represented as a scaling vector and a biased vector. This operation could be considered as a 1x1 convolution with a group size equal to the channel size. From this perspective, AdaIN is a constrained case of the proposed weight generation scheme, since our scheme can generate a convolutional kernel with group size equal to 1 and kernel size larger than 1x1. Moreover, the proposed scheme can be easily combined with the SPADE module. Specifically, we use the proposed generation scheme to generate weights for the SPADE layers, which in turn generate spatially adaptive de-modulation parameters. To justify the importance of weight generation, we compare with AdaIN both using weighted average and our attention module (Table A). We also compare with AdaIN when different dataset sizes are used. The assumption is that when the dataset is small, both methods are able to catch the diversity in the dataset. However, as the dataset size grows larger, AdaIN starts to fail since the expressibility is limited, as shown in Fig. D.

Limitations. Although our network can, in principal, generalize to unseen domains, when the test domain is too different from the training domains it will not perform well. For example, when testing on CG characters which look very different from real-world people, the network struggles. We will include several failure examples in the revised version.

R1: How many examples (the K parameter) are needed? Although we demonstrated that more example images are helpful (Fig. 7b in the paper), our method can work well when $K = 1$. We note that comparisons and examples shown in the paper are using $K = 1$ (i.e., without the attention mechanism).

R1: Comparison to the baselines and importance of weight generation. As explained above, for all comparisons our method only uses a single example image, which is the same as the baselines, so the comparisons are fair. This shows that weight generation (E_c) is useful for generating good quality results. We will make this clear in the revised version.

R2: Novelty. To our best knowledge, we are the first to show that weight generation can be used to solve the adaptive video-to-video synthesis problem. As recognized by R3, this will likely have a wide impact since the topic is of wide interest and our method is generalizable. We will also fix the notations in the revised version.

R3: Comparisons to Zhou et al. 2019 and [7]. We will cite and discuss these papers. Note that both these methods still require *different* models for different persons, which has the same drawback as `vid2vid`. For example, they typically need minutes of training data and days of training time, while our method only needs one image and negligible time for weight generation. Moreover, since code of these methods are not released, we show comparisons to `vid2vid` in Table B for a specific person. We find that our model renders comparable results even when $K = 1$. Moreover, if we further finetune our model based on the example images, we can achieve comparable or even better performance.

R3: Dataset. The dataset is collected by the authors. Different videos are randomly divided into training/test sets. Each video is further divided into clips to ensure each clip only contains one person and does not contain any scene transition, which in general results in 30-1000 frames per clip. We will release the dataset for facilitating research in the field.

R3: Human evaluation. For each pair of comparisons, we generate 100 clips, each of them viewed by 60 workers. Orders are randomized. Hence, each user preference score is computed based on the 6000 evaluations. As reported in the main paper, our preference scores are significantly higher than 0.5. We will also add the citation to the segmentation network.