Algorithm 1: Train a GAN. To use AdvAs, execute the two blue lines instead of the red line.

for *number of training iterations* **do**
  for *number of adversary update steps* **do**
    Estimate adversary loss $L_{\text{adv}}(\theta, \phi)$
    Update adversary parameters ϕ with gradient descent along $\boldsymbol{\nabla}_\phi L_{\text{adv}}(\theta, \phi)$
  end
  Estimate generator loss $L_{\text{gen}}(\theta, \phi)$
  if *use AdvAs* **then**
    Estimate AdvAs loss $r(\theta, \phi)$
    $g = \text{NormalizeGradient}(\boldsymbol{\nabla}_\phi L_{\text{gen}}(\theta, \phi), \boldsymbol{\nabla}_\phi r(\theta, \phi))$
  else
    $g = \boldsymbol{\nabla}_\phi L_{\text{gen}}(\theta, \phi)$
  end
  Update generator parameters θ with gradient descent along g
end

A THE ADVAS ALGORITHM

Algorithm 1 provides pseudocode for training a GAN. Using AdvAs involves estimating an additional loss term for the generator and, to avoid setting the hyperparameter γ, performing normalization of the gradient as described in Section 4.3.

B ESTIMATOR FOR THE ADVAS LOSS

We can derive an alternative unbiased estimate of $r(\theta, \phi)$ to that presented in Section 4.2. To do this, we follow Nowozin et al. (2016) and note that the objective can generally be written in the form

$$h(\theta, \phi) = \mathbb{E}_{\boldsymbol{x}_1 \sim p_\theta, \boldsymbol{x}_2 \sim p_{\text{true}}}[g(\phi, \boldsymbol{x}_1, \boldsymbol{x}_2)]. \tag{12}$$

Therefore,

$$r(\theta, \phi) = \left\| \mathbb{E}\left[\boldsymbol{\nabla}_\phi g(\phi, \boldsymbol{x}_1, \boldsymbol{x}_2)\right] \right\|_2^2, \tag{13}$$

Using ∂_i to denote the ith partial derivative of ϕ, we have that,

$$\left\| \mathbb{E}\left[\boldsymbol{\nabla}_\phi g(\phi, \boldsymbol{x}_1, \boldsymbol{x}_2)\right] \right\|_2^2 = \sum_i \mathbb{E}\left[\partial_i g(\phi, \boldsymbol{x}_1, \boldsymbol{x}_2)\right]^2. \tag{14}$$

As the sum of unbiased estimators is itself unbiased, we need only find an unbiased estimate of $\mathbb{E}\left[\partial_i g(\phi, \boldsymbol{x}_1, \boldsymbol{x}_2)\right]^2$, which is of the form $\mathbb{E}\left[x\right]^2$. Using the identity $\text{Var}(x) = \mathbb{E}\left[x^2\right] - \mathbb{E}\left[x\right]^2$, we find, using Monte Carlo estimates of the expectations, that,

$$\mathbb{E}\left[\partial_i g(\phi, \boldsymbol{x}_1, \boldsymbol{x}_2)\right]^2 = \text{Var}(\partial_i g(\phi, \boldsymbol{x}_1, \boldsymbol{x}_2)) + \mathbb{E}\left[\left(\partial_i g(\phi, \boldsymbol{x}_1, \boldsymbol{x}_2)\right)^2\right] \tag{15}$$

$$\approx \frac{1}{N-1} \sum_{n=1}^{N} \left(\partial_i g(\phi, \boldsymbol{x}_1^{(n)}, \boldsymbol{x}_2^{(n)}) - \overline{\partial_i g}\right)^2 + \frac{1}{N} \sum_{n=1}^{N} \left(\partial_i g(\phi, \boldsymbol{x}_1^{(n)}, \boldsymbol{x}_2^{(n)})\right)^2 \tag{16}$$

$$\equiv \tilde{e}_i(\phi), \tag{17}$$

where $\boldsymbol{x}_1^{(n)} \sim p_\theta, \boldsymbol{x}_2^{(n)} \sim p_{\text{true}}$ and $\overline{\partial_i g} = \frac{1}{N} \sum_{n=1}^{N} \partial_i g(\phi, \boldsymbol{x}_1^{(n)}, \boldsymbol{x}_2^{(n)})$. As $\tilde{e}(\phi)$ is an unbiased estimator it follows that

$$\sum_i \tilde{e}_i(\phi) \approx \left\| \mathbb{E}\left[\boldsymbol{\nabla}_\phi g(\phi, \boldsymbol{x}_1, \boldsymbol{x}_2)\right] \right\|_2^2 \tag{18}$$

is also an unbiased estimator. However, calculating $\sum_i \tilde{e}_i(\phi)$ is currently inefficient since it involves estimating the variance of gradients, which is poorly supported by the implementation of automatic

differentiation in e.g. PyTorch. Therefore, we instead use the following biased estimate of $r(\theta, \phi)$, by directly taking the norm of the Monte Carlo estimate of the expectation in Eq. (13),

$$\tilde{r}(\theta, \phi) \approx \left\| \frac{1}{N} \sum_{n=1}^{N} \nabla_{\phi} g(\phi, \boldsymbol{x}_1^{(n)}, \boldsymbol{x}_2^{(n)} \right\|_2^2, \tag{19}$$

where $\boldsymbol{x}_1^{(n)} \sim p_\theta, \boldsymbol{x}_2^{(n)} \sim p_{\text{true}}$. We emphasize again that comparisons between the unbiased estimate in Eq. (18) and the biased estimate in Eq. (19), did not suggest a significant difference in performance.

C Additional experimental results

C.1 CIFAR10

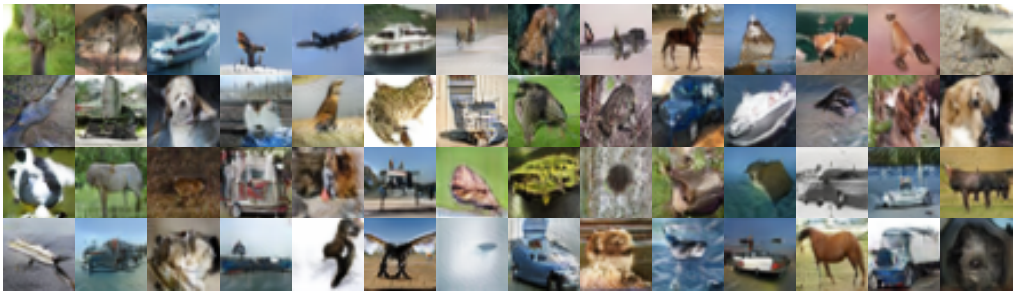

Figure 4: Uncurated samples from the baseline AutoGAN with $n_{\text{adv}} = 5$ after finishing training.

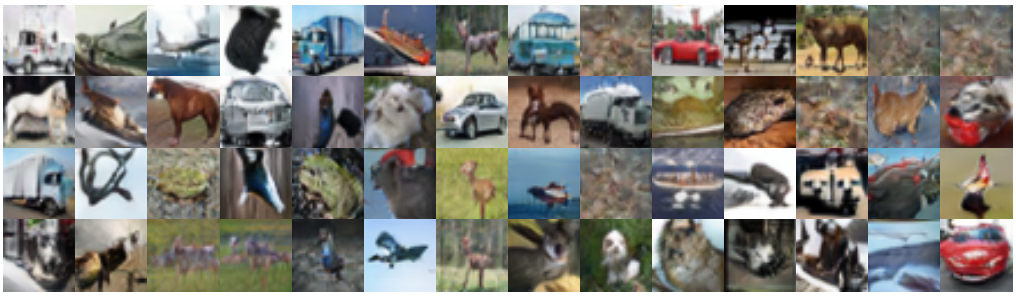

Figure 5: Uncurated samples from the AutoGAN with AdvAs and $n_{\text{adv}} = 5$ after finishing training.

In Figs. 4 and 5, we present a random sample of CIFAR10 images from networks trained with the best-performing values of n_{adv} with and without AdvAs. The networks further use exponential moving averages of their weights. Reflecting the similar FID scores on convergence, there is no obvious qualitative difference between the samples.

C.2 CelebA

Quantitative metrics In Fig. 3 of the main paper, we plot FID scores for CelebA throughout training, evaluated with 1000 samples and an exponential moving average of the generator weights. In Fig. 6 we show corresponding results without using this average of generator weights. The results are qualitatively similar, but the FID scores are slightly higher for each method. Additionally, in the right columns of Figs. 6 and 7, we plot a FID score computed less frequently using 202 599 samples from the generator; this is as many samples as there are images in the CelebA dataset.

Random samples In addition to reporting FID scores for each method in the paper, we provide a random sample of images from a GAN trained with AdvAs (Fig. 9), and one trained without

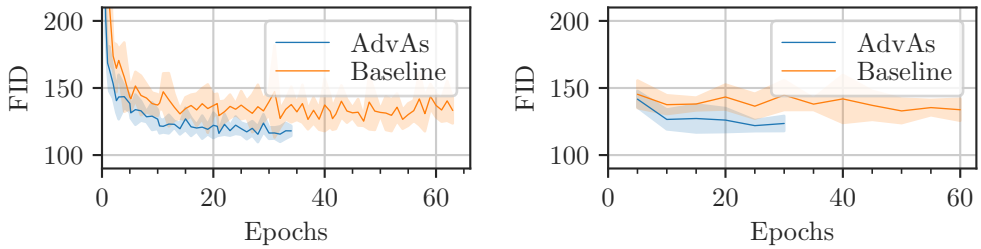

Figure 6: FID scores for StyleGAN2 on CelebA-64 calculated with 1000 samples (left), or 202 599 (right) without an exponential moving average of the weights.

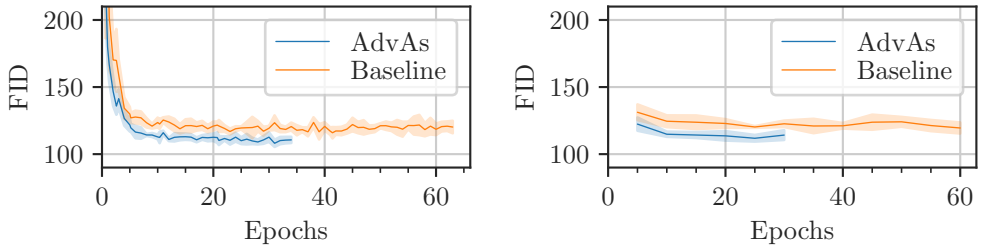

Figure 7: FID scores for StyleGAN2 on CelebA-64 calculated with 1000 samples (left), or 202 599 (right) with an exponential moving average of the weights as in Fig. 3.

(Fig. 10). We also show interpolations between randomly sampled latent variables. If images change smoothly as the latent variables change, this indicates that the network does not overfit to the training data (Brock et al., 2019). The interpolations of networks trained with AdvAs are at least as smooth as those without.

C.3 Training with AdvAs alone

As mentioned in Section 4.1, for some GAN objectives the AdvAs term $r(\theta, \phi)$ has a global minimum for any value of ϕ when $p_\theta = p_{\text{true}}$. A natural question is therefore whether the generator can be fitted to the data purely by minimizing $r(\theta, \phi)$. Figure 8 shows that a generator can be learned in this way for MNIST, although with results that are clearly inferior to minimizing the standard generator objective. This indicates that, although AdvAs can be used to directly match the generated and true distributions, the primary beneficial effect on training comes from using it to "assist" an adversary in addition to using the standard generator loss.

In the figure, the adversary is trained using its standard loss. This is theoretically unnecessary since setting $p_\theta = p_{\text{true}}$ will minimise $r(\theta, \phi)$ for any ϕ. Therefore, ϕ could be sampled at initialization and trained no further. However, we found that training ϕ along with θ was necessary or the images produced would be qualitatively significantly worse than those in Fig. 8. The architecture used is different to other experiments, using bilinear upsampling rather than transposed convolutions. This was necessary to achieve the results shown.

D Proofs

D.1 Proof of Theorem 1

This theorem is based on Theorem 1 of Milgrom & Segal (2002), which assumed $\theta \in [0, 1]$. We generalize this to $\theta \in \mathbb{R}^n$, as Milgrom & Segal noted was possible but did not explicitly prove. The proof is also similar to that of Danskin's theorem (Bertsekas, 1997, p. 717). However, Danskin's theorem makes different assumptions which do not guarantee that $\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta)$ exists, and so is in general limited to considering directional derivatives.

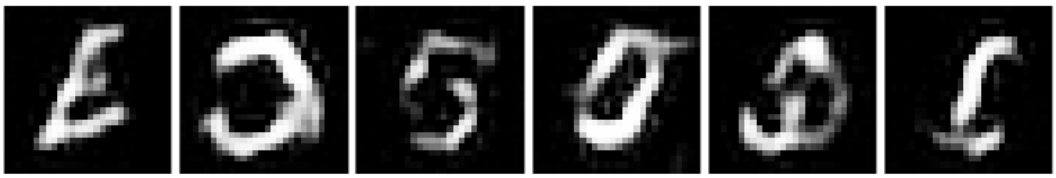

Figure 8: Uncurated samples from a generator trained to minimize $r(\theta, \phi)$ alone. The adversary is trained using its standard loss, $L_{\text{adv}}(\theta, \phi)$. We use the WGAN objective (Eq. (31)) with weight clipping to enforce the Lipschitz constraint, and train for 100 000 iterations. The neural network architecture used was similar to that described in Appendix E.1, but with upsampling rather than transposed convolutions, and batch normalization rather than instance normalization. The samples exhibit some features of MNIST images, but are mostly not recognizable digits.

Let $\theta, \bar{\theta} \in \mathbb{R}^n$ and $\phi^* \in \Phi^*(\theta)$. Then, by the definition of $\mathcal{M}$ in Eq. (5), we have that

$$\mathcal{M}(p_\theta) = h(p_\theta, a_{\phi^*}), \tag{20}$$

$$\mathcal{M}(p_{\bar{\theta}}) \geq h(p_{\bar{\theta}}, a_{\phi^*}), \tag{21}$$

and subsequently

$$\mathcal{M}(p_{\bar{\theta}}) - \mathcal{M}(p_\theta) \geq h(p_{\bar{\theta}}, a_{\phi^*}) - h(p_\theta, a_{\phi^*}). \tag{22}$$

We set $\bar{\theta} = \theta + h \cdot \nu$, for some positive scalar h and $\nu \in \mathbb{R}^n$. Then dividing both sides by h gives

$$\frac{\mathcal{M}(p_{\theta + h\cdot\nu}) - \mathcal{M}(p_\theta)}{h} \geq \frac{h(p_{\theta+h\cdot\nu}, a_{\phi^*}) - h(p_\theta, a_{\phi^*})}{h}. \tag{23}$$

Taking the limit of both sides as $h \to 0$ results in directional derivatives in the direction ν:

$$\boldsymbol{\nabla}^\nu_\theta \mathcal{M}(p_\theta) \geq \boldsymbol{\nabla}^\nu_\theta h(p_\theta, a_{\phi^*}). \tag{24}$$

Due to the Theorem 1's stated restriction that $\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta)$ and $\boldsymbol{\nabla}_\theta h(p_\theta, a_{\phi^*})$ both exists, the directional derivatives are equal to the dot product of ν and these derivatives:

$$\nu \cdot \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) \geq \nu \cdot \boldsymbol{\nabla}_\theta h(p_\theta, a_{\phi^*}). \tag{25}$$

Since this is true for all $\nu \in \mathbb{R}^d$, both of the following must true for all $u \in \mathbb{R}^d$:

$$u \cdot \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) \geq u \cdot \boldsymbol{\nabla}_\theta h(p_\theta, a_{\phi^*}), \quad -u \cdot \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) \geq -u \cdot \boldsymbol{\nabla}_\theta h(p_\theta, a_{\phi^*}). \tag{26}$$

The only way that these can both be true is if the inequality is in fact an equality:

$$\nu \cdot \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) = \nu \cdot \boldsymbol{\nabla}_\theta h(p_\theta, a_{\phi^*}). \tag{27}$$

Finally, this can only be true for all $\nu \in \mathbb{R}^d$ if the gradients themselves are equal, leading to the result

$$\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) = \boldsymbol{\nabla}_\theta h(p_\theta, a_{\phi^*}) \tag{28}$$

and concluding the proof. □

D.2 Proof of Corollary 1

Let an adversary constructor f be a differentiable function with respect to θ such that the *total derivative* of $h(p_\theta, a_{f(\theta)})$ is,

$$\boldsymbol{\nabla}_\theta h(p_\theta, a_{f(\theta)}) = D_1 h(p_\theta, a_{f(\theta)}) + D_2 h(p_\theta, a_{f(\theta)})^{\mathrm{T}} \boldsymbol{J}_\theta(f), \tag{29}$$

From the definition of f, we have that $h(p_\theta, a_{f(\theta)}) = \mathcal{M}(p_\theta)$ for all $\theta \in \Theta$. From the conditions of Theorem 1, $\mathcal{M}(p_\theta)$ is differentiable with respect to θ. Therefore, $\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) = \boldsymbol{\nabla}_\theta h(p_\theta, a_{f(\theta)})$. Writing out the total derivative as in Eq. (29) gives

$$\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) = D_1 h(p_\theta, a_{f(\theta)}) + D_2 h(p_\theta, a_{f(\theta)})^{\mathrm{T}} \boldsymbol{J}_\theta(f). \tag{30}$$

Theorem 1 says that $\boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) = D_1 h(p_\theta, a_{f(\theta)})$. Subtracting this from both sides of the above equation leads to the stated result. □

D.3 PROOF THAT MINIMIZING $r(\theta, \phi)$ REACHES OPTIMUM FOR WGAN

To prove that the assumption is correct for the WGAN, we consider its objective

$$h_{\mathrm{WGAN}}(p_\theta, a_\phi) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{true}}}\left[a_\phi(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p_\theta}\left[a_\phi(\boldsymbol{x})\right], \quad \text{s.t. } \left\|a_\phi\right\|_L \leq 1. \tag{31}$$

Consider $p_\theta = p_{\mathrm{true}}$. Clearly, this will cause $h_{\mathrm{WGAN}}(p_\theta, a_\phi)$ to be zero for any adversary a_ϕ. In turn, this means that any adversary is optimal since $\boldsymbol{\nabla}_\phi L_{\mathrm{adv}}(\theta, \phi) = 0$ and so $r(\theta, \phi) = 0$ for all $\phi \in \Phi$. For any ϕ, this is a global minimum of $r(\theta, \phi)$ with respect to θ. This indicates that AdvAs promotes p_θ to be equal to p_{true} even when the adversary is sub-optimal. Exactly the same analysis can be applied to various GANs with objectives that can be written similarly as a difference between expectations of the adversary's output (Li et al., 2015; Unterthiner et al., 2017).

D.4 PROOF THAT ADVAS PRESERVES CONVERGENCE RESULTS

Proposition. Making the optimal adversary assumption (Section 2.1), ϕ is updated at each training iteration to be a member of $\Phi^*(\theta)$, resulting in an optimal adversary. Then, whether θ is updated to decrease $L_{\mathrm{gen}}^{\mathrm{AdvAs}}(\theta, \phi)$ or $L_{\mathrm{gen}}(\theta, \phi)$, the fixed points to which the generator can converge are the same.

Proof. Having made the optimal adversary assumption, these two different generator losses have the gradients

$$\boldsymbol{\nabla}_\theta L_{\mathrm{gen}}(\theta, \phi) = \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) \tag{32}$$

and

$$\boldsymbol{\nabla}_\theta L_{\mathrm{gen}}^{\mathrm{AdvAs}}(\theta, \phi) = \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) + \lambda \cdot \boldsymbol{\nabla}_\theta r(\theta, \phi) \quad \text{where} \quad \phi \in \Phi^*(\theta). \tag{33}$$

$$= \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) + \lambda \cdot \boldsymbol{\nabla}_\theta \left\|\boldsymbol{\nabla}_\phi L_{\mathrm{adv}}(\theta, \phi)\right\|_2^2 \tag{34}$$

$$= \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta) + \lambda \cdot 2\big(\underbrace{\boldsymbol{\nabla}_\phi L_{\mathrm{adv}}(\theta, \phi)}_{\text{adversary's gradient}}\big) \cdot \big(\boldsymbol{\nabla}_\theta \boldsymbol{\nabla}_\phi L_{\mathrm{adv}}(\theta, \phi)\big). \tag{35}$$

As long as $\boldsymbol{\nabla}_\theta \boldsymbol{\nabla}_\phi L_{\mathrm{adv}}(\theta, \phi)$ exists, a sufficient condition for the gradients of $\boldsymbol{\nabla}_\theta L_{\mathrm{gen}}(\theta, \phi)$ and $\boldsymbol{\nabla}_\theta L_{\mathrm{gen}}^{\mathrm{AdvAs}}(\theta, \phi)$ to be equal is therefore if the adversary's gradient term marked in Eq. (35) is $\mathbf{0}$. We will now show that this is the case. To do so, we invoke Fermat's theorem on stationary points, which states that at least one of the following is true for each $\phi' \in \Phi^*(\theta)$:

1. ϕ' is in the boundary of Φ.
2. $h(p_\theta, a_{\phi'})$ is not differentiable at ϕ'.
3. $\boldsymbol{\nabla}_{\phi'} L_{\mathrm{adv}}(\theta, \phi') = \mathbf{0}$.

As long as items 1 and 2 are not the case, we have that $\boldsymbol{\nabla}_\phi L_{\mathrm{adv}}(\theta, \phi) = \mathbf{0}$. Therefore,

$$\boldsymbol{\nabla}_\theta L_{\mathrm{gen}}^{\mathrm{AdvAs}}(\theta, \phi) = \boldsymbol{\nabla}_\theta \mathcal{M}(p_\theta). \tag{36}$$

Since, under the stated assumptions, both losses have the same gradients with respect to θ, the sets of fixed points to which the parameters can converge under each are the same. □

E EXPERIMENTAL SETUP

E.1 MNIST

We normalize the data to lie between 0 and 1. The generator we use takes as input a random, unit Gaussian-distributed, vector of dimension 128. This is mapped by a fully-connected layer with a ReLU activation to a 7×7 tensor with 64 channels. We then have two blocks each consisting of a transposed convolution, each halving the number of channels, followed by instance normalization and a ReLU activation. Following the two blocks, there is a final transposed convolution with a sigmoid activation. The discriminator mirrors this structure: it uses three blocks each consisting of a convolution (to 64, 128, and 256 channels respectively) followed by instance normalization and a ReLU activation function. Finally a fully-connected layer maps the resulting tensor to a scalar. See the code for full details. We train the GAN with a batch size of 256; a learning rate of 10^{-3} for both the generator and the discriminator; and the Adam optimizer with hyperparameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Each experiment is performed on a single NVIDIA V100 GPU (32G HBM2 memory) and Intel Silver 4216 Cascade Lake @ 2.1GHz CPUs.

E.2 CIFAR10

We use the best-performing architecture reported by Gong et al. (2019). All hyperparameters are the same except for n_{adv}, which we vary. We use the officially released code[3]. Each GAN is trained on a single NVIDIA V100 GPU (32G HBM2 memory) and Intel Silver 4216 Cascade Lake @ 2.1GHz CPUs.

E.3 CELEBA

We use a publicly-available implementation of StyleGAN2[4] with the standard hyperparameter settings recommended by Karras et al. (2020). Each experiment is performed on a single NVIDIA P100 GPU (12G HBM2 memory) and Intel E5-2650 v4 Broadwell @ 2.2GHz CPUs.

[3] https://github.com/VITA-Group/AutoGAN

[4] https://github.com/lucidrains/stylegan2-pytorch

Figure 9: Uncurated samples and interpolations from our baseline StyleGAN2 trained for 140 hours.

Figure 10: Uncurated samples and interpolations from StyleGAN2 with AdvAs trained for 140 hours.