GAN Cocktail: mixing GANs without dataset access

Omri Avrahami[1], Dani Lischinski[1], and Ohad Fried[2]

[1]The Hebrew University of Jerusalem [2]Reichman University

Abstract. Today's generative models are capable of synthesizing high-fidelity images, but each model specializes on a specific target domain. This raises the need for model merging: combining two or more pre-trained generative models into a single unified one. In this work we tackle the problem of model merging, given two constraints that often come up in the real world: (1) no access to the original training data, and (2) without increasing the network size. To the best of our knowledge, model merging under these constraints has not been studied thus far. We propose a novel, two-stage solution[1]. In the first stage, we transform the weights of all the models to the same parameter space by a technique we term model rooting. In the second stage, we merge the rooted models by averaging their weights and fine-tuning them for each specific domain, using only data generated by the original trained models. We demonstrate that our approach is superior to baseline methods and to existing transfer learning techniques, and investigate several applications.

Keywords: Generative Adversarial Networks, Model Merging

1 Introduction

Generative adversarial networks (GANs) [9] have achieved impressive results in neural image synthesis [5, 13, 15, 16]. However, these generative models typically specialize on a specific image domain, such as human faces, kitchens, or landscapes. This is in contrary to traditional computer graphics, where a general purpose representation (e.g., textured meshes) and a general purpose renderer can produce images of diverse object types and scenes. In order to extend the applicability and versatility of neural image synthesis, in this work we explore *model merging* — the process of combining two or more generative models into a single conditional model. There are several concrete benefits to model merging:

1. It is well suited for decentralized workflows. Different entities can collect their own datasets and train their own models, which may later be merged.
2. If performed properly, merged models can reduce memory and computation requirements, enabling their use on edge devices with limited resources.
3. Merged models enable semantic editing across domains, as described next.

[1] Code is available at: https://omriavrahami.com/GAN-cocktail-page/

GANs often produce a semantically meaningful latent space. Several embedding techniques [1,2,34] have been proposed to map real input images to latent codes of a pre-trained GAN generator, which enables semantic manipulation. Images can be interpolated and transformed using semantic vectors in the embedding space [11, 39], effectively using it as a strong regularizer. A problem arises when one wants to use several pre-trained generators for semantic manipulations (e.g., interpolating between images from GAN A and GAN B) — the different models do not share the same latent representation, and hence do not "speak the same language". Model merging places several GANs in a shared latent space, allowing such cross-domain semantic manipulations.

We tackle the problem of merging several GAN models into a single one under the following real-world constraints:

1. **No access to training data.** Many GAN models are being released without the data that they were trained on. This can occur because datasets are too large [6,35,36] or due to privacy/copyright issues. Hence, we assume that no training data is available, and only rely on data generated by the pre-trained models.
2. **Limited computing power.** A naïve approach to merging several GAN models is to sample from them separately (e.g., by multinomial sampling functions [45]). With this approach, the model size and inference time grow linearly with the number of GAN models, which may not be practical due to lack of computing power (e.g., edge devices). In addition, this approach does not result in a shared latent space, so it does not support cross-domain semantic manipulations as described earlier. Our goal is to maintain a constant network size, regardless of the number of GANs being merged.

To the best of our knowledge, performing model merging under these constraints has not been studied yet. This is a challenging task: pre-trained GANs typically do not model the entire real image distribution [4]; hence, learning from the outputs of pre-trained models will be sub-optimal. In addition, the constraint on the model size may reduce its capacity.

We start by adapting existing solutions from the field of transfer-learning as baselines (Section 3). Next, we present our novel two-stage solution for model merging. We first transfer the weights of the input models to a joint semantic space using a technique we term model rooting (Section 4.1). We then merge the rooted models via averaging and fine-tuning (Section 4.2). We find that model rooting introduces an inductive bias that helps the merged model achieve superior results compared to baselines and to existing transfer-learning methods (Section 5).

To summarize, this paper has the following contributions:

- We introduce the real-world problem of merging several GAN models without access to their training data and with no increase in model size or inference time.
- We adapt several transfer-learning techniques to the GAN merging problem.
- We introduce a novel two-stage approach for GAN merging and evaluate its performance.

2 Related Work

Generative adversarial networks: GANs [9] consist of a generator G and a discriminator D that compete in a two-player minimax game: the discriminator tries to distinguish real training data from generated data, and the generator tries to fool the discriminator. Training GANs is difficult, due to mode collapse and training instability, and several methods focus on addressing these problems [10, 22, 23, 26, 37], while another line of research aims to improve the architectures to generate higher quality images [5, 13, 15, 16]. Karras et al. [15, 16] introduced the StyleGAN architecture that leads to an automatically learned, unsupervised separation of high-level attributes and stochastic variation in the generated images and enables intuitive, scale-specific control over the synthesis. For our experiments we use the StyleGAN2 framework. It is important to note that our approach, as well as the baselines, are model-agnostic and there is no dependency on any StyleGAN-specific capabilities. We demonstrate mixing between models of different architectures in Supp. Section 2.1.

Transfer learning: Learning how to transfer knowledge from a source domain to a target domain is a well-studied problem in machine learning [3, 7, 19, 29, 31, 32, 43], mainly in the discriminative case. It is important to note that the transfer-learning literature focuses on the case where there is a training dataset for the target domain, which is not the case in our scenario. Recent works that are more related to our problem by Shu et al. [41] and Geyer et al. [8] demonstrate the ability to perform transfer learning from several source models into a single target model. However they are not applicable to our setting because: (1) neither method tackles generative models, as we do; (2) both of these methods assume that all the source models share the same architecture, whereas our problem formulation specifically focuses on the general case with arbitrary architectures (which is the real-world scenario, especially for generative models); (3) both methods assume access to training data, while we assume that the training data is unavailable; (4) T-IMM method [8] assumes that the user trains the *source models* incrementally, which is different from our setting, where the source models training is not under the user control. To conclude, the current literature mainly focuses on the discriminative case and assumes access to the training data.

As shown by Wang et al. [46], the principles of transfer learning can be applied to image generation with GANs. Later, Noguchi and Harada [30] proposed to constrain the training process to only update the normalization parameters instead of all of the model's trainable parameters. This shrinks the model capacity, which mitigates the overfitting problem and enables fine-tuning with an extremely small dataset. However, limiting the capacity of the model enables to only change the style of the objects but not their shape, which isn't applicable to our setting, where the merged image domains may exhibit objects of completely different shapes.

Another approach for GAN transfer learning consists of adding a layer that steers the generated distribution towards the target distribution, which is also applicable for sampling from several models [45]. However, this approach stitches

the source models together, and thus the model size and the inference time grow linearly in the number of source models. In addition, the models in this approach do not share the same latent space which limits their applicability.

Continual learning: Continual Learning, also known as lifelong learning, is a setting where a model learns a large number of tasks sequentially without forgetting knowledge obtained from the preceding tasks, even though their training data is no longer available while training newer ones. Continual learning mainly deals with the "catastrophic forgetting" phenomenon, i.e., learning consecutive tasks without forgetting how to perform previously trained ones. Most previous efforts focused on classification tasks [18, 21, 50], and were later also adapted to the generative case [38].

Again, the literature focuses on the case where the new dataset is available, while the old dataset is not, which is not the case in our scenario, but we can adapt it to our setting, and do so in Section 3.4. Also, approaches that rely on designated architectures (e.g., lifelong GANs [51]) cannot be adapted to our setting.

Another approach addressing the catastrophic forgetting problem that was proposed by Wu et al. [47] is a memory-replay mechanism that uses the old generative model as a proxy to the old data. Our adaptation of the method of Wang et al. [46] to our setting in Section 3.3 may be viewed also as adapting the approach of Wu et al. [47].

3 Problem Formulation and Baselines

Our goal is to merge several GANs without access to their original training data. For example, given two trained GAN models, one that generates images of cats and another that generates images of dogs, we want to train a new single GAN model that produces images from both domains, without increasing the model size. Below, we first formulate the problem, present several baselines, and then introduce our approach to solving this task in Section 4.

3.1 Problem Formulation

We are given N GANs: $\{GAN_i = (G_i, D_i)\}_{i=1}^{N}$, where GAN_i is pretrained on dataset $data_i$ and consists of a generator G_i and a discriminator D_i. We denote the distribution of images that are produced by the generator G_i by $P_{G_i}(z)$ and the real data distribution as $P_{data_i}(x)$.

Our goal is to create a "union GAN", $\textsc{UnionGAN} = (G_u, D_u)$, which is a conditional GAN [25], with the condition c indicating which of the N domains the generated sample should come from: $\forall_{c\in[N]} P_{G_u}(z, c) = P_{data_c}(x)$ and $P_{D_u}(x, c)$ is the probability that x came from $data_c$, rather than P_{G_u}. Note that the datasets $data_i$ are not provided. Furthermore, the N pretrained GAN models may have different architectures. Below, we adapt some current techniques from the transfer learning literature to address this problem.

Table 1: Comparison between FID scores of models that were trained on real and generated images

	Trained on	
Dataset	real	generated
FFHQ	**5.58**	8.84
LSUN cat	**17.37**	21.78
LSUN dog	**20.48**	24.31
LSUN car	**7.12**	12.79

3.2 Baseline A: Training From Scratch

Arguably, the simplest approach would be to train UnionGAN from scratch, by using the samples generated by the pretrained input GANs as the only training data. The objective function of a two-player minimax game that we aim to solve in this case is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{c\in[N],z\sim p_z(z),x\sim P_{G_c}(z)}[\log D(x,c)] + \mathbb{E}_{c\in[N],z\sim p_z(z)}[\log(1-D(G(z,c),c))] \quad (1)$$

Note that this formulation differs from a standard GAN in two ways: the discriminator is trained on the outputs of the given generators instead of on real data, and UnionGAN is conditioned on both the class and the latent code z. Thus, we simply treat the pre-trained generators as procedural sources of training data. We convert the unconditional model into a conditional one by adding a class embedding layer to the generator and concatenate its output the the latent code z. An embedding layer is also added to the discriminator. See the supplementary material for more details.

Although the number of generated images that can be produced by a generator is unlimited (in contrast to a real training dataset), we found that training using the real dataset produces better results. This is likely caused by the fact that the pretrained GANs generate only a subset of the training data manifold. To validate that the issue is not due to limited capacity, we train UnionGAN on the degenerate case of $N = 1$, using different pretrained GANs, and observe a consistent increase in the resulting FID score, as reported in Table 1. In general, the best results can be achieved using the original real training data.

3.3 Baseline B: TransferGAN

The above method uses only the outputs of the pretrained models, thereby using them as black boxes. Below we improve this method by using not only the generated data, but also the weights of the trained models.

Specifically, we adapt TransferGAN [46] to our problem as follows: we initialize the UnionGAN with the i-th source model, and then train it on the outputs

of all the GAN models (as described in the previous section) until convergence. Thus, we treat one of the models as both an initializer and a data source, and the remaining models as training data sources.

Compared to training from scratch, such initialization lowers the total FID score (for the union of the datasets), as reported in Table 3. Furthermore, Table 4 shows that the FID score is lowered not only for the i-th dataset, but for the other datasets as well.

3.4 Baseline C: Elastic Weight Consolidation

We observe that although the TransferGAN approach improves the final FID score, if we focus on the source model, we can see that its FID score (on the source class) is initially high and is degraded over the training process (the FID score of the original dataset class increases while the FIDs for the other classes decrease). This occurs due to catastrophic forgetting [18] and can be mitigated by Elastic Weight Consolidation (EWC) [18, 20], applied to TransferGAN.

In order to assess the importance of the model parameters to its accuracy, we use Fisher information, which formulates how well we estimate the model parameters given the observations. In order to compute the empirical Fisher information given a pretrained model for a parameter θ_i, we generate a certain amount of data X and compute: $F_i = \mathbb{E}[(\frac{\partial}{\partial \theta_i}\mathcal{L}(x|\theta_i))^2]$ where $\mathcal{L}(x|\theta_i)$ is the log-likelihood. In the generative case, we can equivalently compute the binary cross-entropy loss using the outputs of the discriminator that is fed by the outputs of the generator.

Thus, feeding the discriminator with the generator's outputs, we generate a large number of random samples, compute the binary cross-entropy loss on them and compute the derivative via back-propagation. We can add to our loss term the Elastic Weight Consolidation (EWC) penalty: $\mathcal{L}_{EWC} = \mathcal{L}_{adv} + \lambda \sum_i F_i(\theta_i - \theta_{S,i})$ where θ_S represents the weights learned from the source domain, i is the index of each parameter of the model and λ is the regularization weight to balance different losses.

Unfortunately, as can be seen in Table 4, this procedure mitigates the catastrophic forgetting phenomena at the expense of degrading performance on the other classes. We also experimented with a more naïve approach of applying a L2 loss on the source model weights, but, as we expected, the results were much worse for all but the source class.

4 Our Approach: GAN Cocktail

The main limitation of the transfer-learning approach is that it only uses the weights of one of the pre-trained GAN models (GAN_i, the source model). In order to leverage the weights of all models, we propose a two-stage approach: At the first stage we perform *model rooting* for all the input GAN models and in the second stage we perform *model merging* by averaging the weights of the rooted models and then fine-tuning them using only data generated by the original models to obtain the merged UnionGAN.

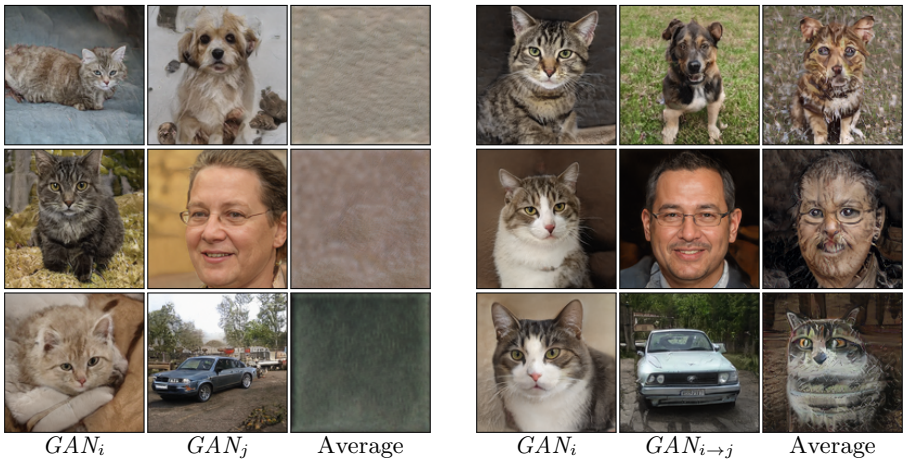


Fig. 1: **Left:** Averaging of two models GAN_i and GAN_j with the same architecture, but without a common root. *Average* is a model in which each weight is the arithmetic mean of the corresponding weights in the original two models. Each row corresponds to the same input z. Note that the resulting images exhibit no obvious semantic structure. **Right:** Averaging of two models which have a common root model. The resulting networks (before any fine-tuning) produce images which are a semantically meaningful mix of the two object categories.

4.1 First stage: Model rooting

Our goal is to merge the GAN models while maintaining as much information and generative performance as possible from the original models. In order to do so we need to somehow combine the weights of these models.

One way to combine several neural networks is by performing some arithmetic operations on their parameters. For example, Exponential Moving Average (EMA) is a technique for averaging the model weights during the training process in order to merge several versions of the same model (from different checkpoints during the training process). EMA may be used for both discriminative [42] and generative tasks [13, 15, 16].

In order to average the weights of several models, the weights must have the same dimensions. However, this condition is not sufficient for achieving meaningful results. For example, Figure 1 (left) demonstrates that if we simply average the weights of two generators with the same architecture, which were trained on two different datasets, the resulting images have no apparent semantic structure.

A key feature in the EMA case is that the averaging is performed on the same model from different training stages. Thus, we can say that the averaging is done on models that share the same *common ancestor* model, and we hypothesize that this property is key to the success of the merging procedure.

Thus, given N source GANs, $\{GAN_i = (G_i, D_i)\}_{i=1}^{N}$, we (a) convert them to the same architecture (if their original architectures differ), and (b) create

Table 2: Rooted vs. not-rooted distance. The distance between the weights of the rooted model is much closer than that of the not-rooted model

Merged Datasets	$d(A, B)$	$d(A, A \to B)$
cat + dog	418.55	**232.21**
FFHQ + cat	433.87	**252.34**
cat + car	454.23	**264.31**

a common ancestor for all the models. To meet these conditions we propose the model rooting technique: we choose one of the models arbitrarily (see Section 5.1 for details) to be our root model GAN_r; next, for each $i \in [N] \setminus r$ we train a model that is initialized by GAN_r on the outputs of GAN_i, with the implicit task of performing catastrophic forgetting [18] of the source dataset r. We denote each one of the resulting models as $GAN_{r \to i}$. Now, models GAN_r and $GAN_{r \to i}$ not only share the same architecture but also share a common ancestor. Hence, averaging their weights will yield more semantically meaningful results, as demonstrated in Figure 1 (right).

In order to quantify the distance between two models GAN_A and GAN_B we can measure the L_2 distance between their weights, i.e.: $d(A, B) = \frac{1}{L} \sum_i \|\theta_{A,i} - \theta_{B,i}\|_2$ where $\theta_{A,i}$ is the i layer of model A, $\theta_{B,i}$ is the i layer of model B, and L is the number of layers. Figure 1 (right) implies that the weights of GAN_A are more aligned with those of $GAN_{A \to B}$ than with the weights of GAN_B. In order to verify this quantitatively, we report the distances $d(A, B)$ and $d(A, A \to B)$ in Table 2. Indeed, the rooted models $GAN_{A \to B}$ are much closer to the root, despite being trained on other datasets until convergence. Note that semantically closer datasets (e.g., cats and dogs) yield a smaller distance.

To conclude, the model rooting step transfers all the models to the same architecture and aligns their weights such that they can be averaged. Next, inspired by EMA, we will show that the averaging of the models introduces an inductive-bias to the training procedure that yields better results.

4.2 Second stage: Model merging

We now have N rooted models, averaging whose weights yields somewhat semantically meaningful results. However, images generated by the averaged models are typically somewhere in between all the training classes (Figure 1, rightmost column). We want the model to learn to reuse filters that are applicable for all datasets, and differentiate the class-specific filters. For that, we continue with an adversarial training of the averaged model using the original GAN models as the data sources.

Specifically, given the N rooted models from the previous stage: GAN_r and $\{GAN_{r \to i}\}_{i \in [N] \setminus r}$ we create an average model: $GAN_a = (G_a, D_a)$, s.t. $\theta_a = \frac{1}{N}(\theta_r + \sum_{i \in [N] \setminus r} \theta_{r \to i})$ where θ_i are the parameters of model i. We also

experimented with more sophisticated weighted average initialization based on the diagonal of the Fisher information matrix [18] but it did not improve the results over a simple averaging. We then fine-tune GAN_a using the outputs of the original GAN_i models to obtain the desired UnionGAN.

5 Results

Our main evaluation metric is the commonly-used Fréchet Inception Distance (FID) [12] which measures the Fréchet distance in the embedding space of the inception network between the real images and the generated images. The embedded data is assumed to follow a multivariate normal distribution, which is estimated by computing their mean and covariance. We measure quality by computing FID between 50k generated images and all of the available training images, as recommended by Heusel et al. [12].

All the FID scores that are reported in this paper were computed against the original training data. Note that this is for evaluation purposes only, and our models did not have access to the original data during training.

We evaluate our model on several representative cases using LSUN [49] and FFHQ [15] datasets. We specifically chose to compare between domains that are semantically close (cats and dogs), as well as domains that are more semantically distant (cats and cars). In addition, we compare aligned and unaligned datasets:

- **Aligned and unaligned images:** we used LSUN cat dataset which contains images of cats in different poses and sizes, and FFHQ dataset which contains images of human faces that are strictly aligned. The FID between these two datasets is 196.59.
- **Unaligned imaged from related classes:** we used LSUN cat and LSUN dog classes. The FID between the two datasets is 72.2.
- **Unaligned imaged from unrelated classes:** we used LSUN cat and LSUN car classes. The FID between the two datasets is 161.62.

The FID distances reported above provide an indication of the semantic proximity between each pair of datasets. Not surprisingly, cat images are semantically closer to dog images than to images of humans (FFHQ) or of cars.

We compare our method against the following methods: training from scratch (Section 3.2), TransferGAN (Section 3.3), Elastic Weight Consolidation (Section 3.4) and the recently proposed Freeze Discriminator method [28] which aims to improve transfer learning in GANs by freezing the highest-resolution layers of the discriminator during transfer.

In Table 3 we calculate the FID score between the union of all classes and the union of 50K samples of each class of the generated images. Our method outperforms other methods on all the datasets we experimented with.

In addition, we evaluated the FID score on each class separately in order to measure the effect of each method on each class. As can be seen in Table 4, when the classes are semantically close, such as in the case of cat + dog, our method achieves better results than all the baselines. When the classes are semantically

Table 3: Comparison of FID score w.r.t. the union of all the datasets, for several dataset combinations. Cat, dog, and car datasets are taken from LSUN [49]

Datasets	FFHQ cat	cat dog	cat car	FFHQ cat dog	FFHQ cat dog car
From scratch	19.61	27.58	20.52	23.22	24.88
TransferGAN [46]	18.63	22.17	17.77	20.64	19.34
EWC [18]	19.45	22.17	17.65	19.47	19.14
Freeze-D [28]	18.17	21.92	17.52	19.71	19.41
Our	**16.44**	**20.77**	**16.85**	**18.98**	**18.44**
Upper bound (real data training)	11.86	17.68	14.28	15.93	16.45

Table 4: FID scores per-class over different dataset combinations

	LSUN cat+dog		**LSUN cat+car**		**LSUN cat+FFHQ**	
Dataset	cat	dog	cat	car	FFHQ	cat
Scratch	30.37	33.21	32.21	14.43	13.35	31.64
TransferGAN [46]	23.32	28.84	30.06	11.49	11.16	32.08
EWC [18]	23.04	30.11	30.65	**10.54**	**9.85**	35.36
Freeze-D [28]	23.36	28.40	29.78	11.44	10.64	31.57
Our	**22.08**	**26.52**	**27.78**	11.59	10.60	**27.82**
Upper bound (real data training)	16.49	24.75	23.23	9.5	8.49	19.19

distant, such as in the case of cat + FFHQ or cat + car, we can see that EWC achieves better results on the class with respect to which it minimizes its weights distances, but this comes at the expense of the other class. This is the reason for the better overall performance of our method, reported in Table 3.

As mentioned at the outset, the premise of this work is that the original training data is not available (which is the case with many real-world models). If the original training data is available to the merging process, the best merging results may unsurprisingly be achieved by simply training the new class-conditioned model on the union of the original training datasets. Results achieved in this manner are an upper bound for the results that can be achieved without access to the training data. Table 3 and Table 4 show the gap between our merging approach (without training data) and the aforementioned upper bound.

5.1 Choosing the root model

At the first stage of our approach we arbitrarily choose one of the models to serve as the root model. This raises the question of whether the choice of the root model

Table 5: Our method outperforms the baselines, in terms of FID score, regardless of the model that is chosen as the root model.

	LSUN cat + LSUN dog		**LSUN cat + FFHQ**	
Root model	LSUN cat	LSUN dog	FFHQ	LSUN cat
Scratch	27.58	27.58	19.61	19.61
TransferGAN [46]	22.17	21.11	18.63	16.40
EWC [18]	22.17	25.16	19.45	16.52
Freeze-D [28]	21.92	25.03	18.17	16.87
Our	**20.77**	**20.03**	**16.44**	**15.60**

matters. Table 5 shows that our method outperforms the baselines regardless of the model that is chosen as the root model. On the other hand, it does not mean that the choice of the root model is insignificant for the overall performance of the final merged model. As we can see from Table 5, when merging LSUN cat and LSUN dog models, the better overall result is achieved when LSUN dog is chosen as the root, while when merging the LSUN cat and FFHQ models, the better result is achieved by choosing LSUN cat to be the root model.

We hypothesized that a better candidate for the root model would be the generator that is more diverse, i.e., whose generated images are semantically far from each other. To test our hypothesis we calculated the diversity by measuring pairwise LPIPS scores between the generated images of each model. However, we found that it is not always the case that the more diverse generator is the better root model.

5.2 Applications

The output of our model is a single conditional GAN with a common latent space for all the classes. Hence, the merged model supports a variety of GAN applications from the literature. To name a few:

Latent space interpolation is used for demonstrating the smoothness of the latent space of a GAN. It can also be used for creating smooth transition sequences between objects of different classes. In Figure 2 we demonstrate a transition between a cat and a dog by interpolating between their two w latent vectors in the merged model from two different classes.

Style mixing, introduced by Karras et al. [15], is the ability the mix between generated images on different semantic levels (e.g., gender, hairstyle, pose, etc.) by feeding a different latent vector w to different generation layers. Given a shared latent space for different classes enables us to use the style mixing mechanism to mix attributes from images belonging to these different classes, e.g., change the pose of a cat to that of a dog, while retaining the appearance of the cat. A few such examples are shown in Figure 3. Note how both the pose and

Fig. 2: Interpolation in the merged model's latent space of between images of different classes.

the general shape are taken from the source class (e.g., the shape of the ears in column 1 is taken from the dog images, rather than from the cat images).

Semantic editing is the ability to perform image editing operations on images by manipulating their latent space [11, 39, 44]. One advantage of our framework is the ability to edit images from different domains using the same latent direction because of the shared latent space. For example, given a model that merges FFHQ and LSUN cat generators, we can leverage an off-the-shelf pose classifier, which is available for humans but not for cats, in order to classify poses as "positive" (pose from left to right) or "negative" (pose from right to left). Applying this classifier only to images of humans generated by the merged model, we obtain a direction in the shared latent space that corresponds to a pose change, as the hyperplane normal of a linear SVM trained on the latent vectors of the human faces. Figure 4 demonstrates that the same latent direction (that was calculated on humans only) can be applied for both humans and cats. So, using our model merging solution we can leverage off-the-shelf classifiers on one class to operate on all of the classes.

6 Limitations and Future Work

Due to our self-imposed constraint on model size, we have found that our solution (and the baselines) are sensitive to the number of source models and their properties: merging more models or merging models with semantically distant distributions produces higher FID scores, as may be seen in Table 6. For example, merging LSUN cat and LSUN dog produced better FID scores on the cat dataset on all the baselines in comparison to merging LSUN cat and LSUN car. We conjecture that this happens because more filters can be reused among the semantically similar datasets. Additionally, we can see that merging four of the datasets produces the worst result on our method. Notice that merging FFHQ + cat + dog produced a better result than cat + car and FFHQ + cat because of the semantical closeness of cat and dog.

Yet another disadvantage of our method is that the training time is longer due to the two-stage approach. The baselines converge faster to their local minima, but result in a higher FID score than our method.

Future work can be to relax the capacity constraint and allow a minimal capacity and run-time increase to enable merging more models or models from semantically distant distributions with better results in terms of FID score.

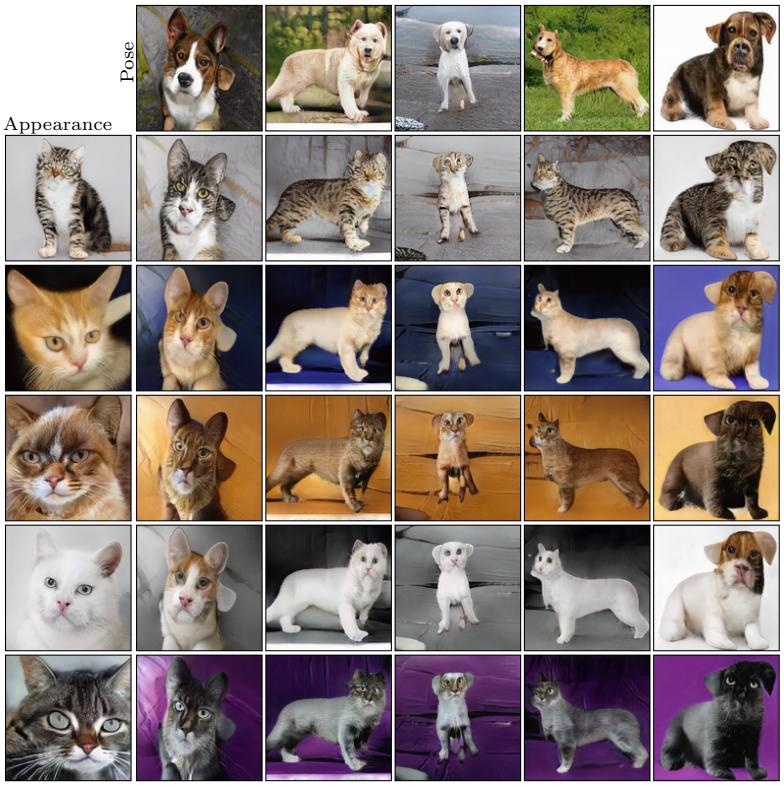


Fig. 3: Style mixing between images of two different domains: taking the pose and shape from the dog image and the appearance from the cat image.

7 Broader Impact

One major barrier when developing a machine learning model is the lack of training data. Many small organizations and individuals find it hard to compete with larger entities due to the lack of training data. This is especially true in fields where curating and annotating the training data is time-consuming and expensive (e.g., medical data). It was shown that GANs can be used in order to augment and anonymize sensitive training data [40, 48]. Our method can be used to alleviate the problem of scarce training data, by allowing entities with small budgets to use pre-trained GANs in two ways: use them as training data, and reuse some of the knowledge incorporated in their weights.

On the other hand, our method may amplify the copyright issues that arise when training a model on synthetic data. The legal implications of training a model using another model that was trained on a private or copyrighted dataset are currently unclear. We would like to encourage the research community to work with governments and legal scholars to establish new laws and regulations in lockstep with the rapid advancement of synthetic media.

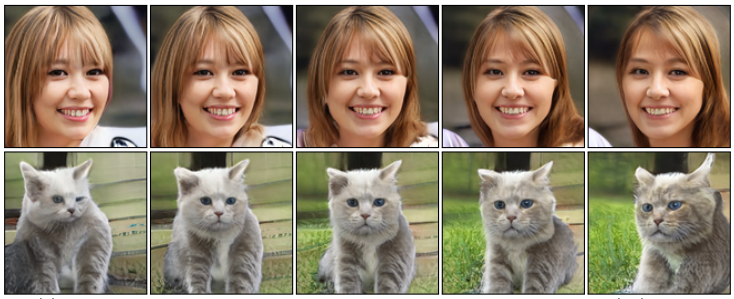


Fig. 4: We determine the pose direction in the latent space of the merged model of FFHQ and LSUN cat using images of the FFHQ class only. Applying this direction to images from both classes reveals that the semantics stay the same.

Table 6: Comparison of FID scores of cat class only, when merging LSUN cat with different datasets. Semantically closer datasets (e.g., cats and dogs) lead to better scores compared to far datasets (e.g., cats and cars). Merging 4 datasets produces the worst result

Datasets	cat dog	cat car	FFHQ cat	FFHQ cat dog	FFHQ cat dog car
From scratch	30.37	32.21	31.64	34.75	45.02
TransferGAN [46]	23.32	30.06	32.08	29.06	30.50
EWC [18]	23.04	30.65	35.36	25.70	27.98
Freeze-D [28]	23.36	29.78	31.57	28.68	30.95
Our	22.08	27.78	27.82	26.80	30.17

8 Conclusions

In this paper, we introduced the problem of merging several generative adversarial networks without having access to the training data. We adapted current methods for transfer-learning and continual-learning and set them as our baselines. We then introduced our novel two-stage solution to the GAN mixing problem: model rooting and model merging. Later, we compared our method to the baselines and demonstrated its superiority on various datasets. Finally, we presented some applications of our model merging technique.

Acknowledgments This work was supported in part by Lightricks Ltd and by the Israel Science Foundation (grants No. 2492/20, 1574/21, and 2611/21).

References

1. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the StyleGAN latent space? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4432–4441 (2019) 2
2. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan++: How to edit the embedded images? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8296–8305 (2020) 2
3. Bao, Y., Li, Y., Huang, S.L., Zhang, L., Zheng, L., Zamir, A., Guibas, L.: An information-theoretic approach to transferability in task transfer learning. In: 2019 IEEE International Conference on Image Processing (ICIP). pp. 2309–2313. IEEE (2019) 3
4. Bau, D., Zhu, J.Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., Torralba, A.: Seeing what a GAN cannot generate. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4502–4511 (2019) 2
5. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2018) 1, 3
6. Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I.: Generative pretraining from pixels. In: International Conference on Machine Learning. pp. 1691–1703. PMLR (2020) 2
7. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: International conference on machine learning. pp. 647–655. PMLR (2014) 3
8. Geyer, R., Corinzia, L., Wegmayr, V.: Transfer learning by adaptive merging of multiple models. In: International Conference on Medical Imaging with Deep Learning. pp. 185–196. PMLR (2019) 3
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc. (2014), https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf 1, 3
10. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of Wasserstein GANs. In: NIPS (2017) 3
11. Härkönen, E., Hertzmann, A., Lehtinen, J., Paris, S.: GANSpace: Discovering interpretable GAN controls. Advances in Neural Information Processing Systems **33** (2020) 2, 12
12. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf 9
13. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017) 1, 3, 7
14. Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., Aila, T.: Training generative adversarial networks with limited data. arXiv preprint arXiv:2006.06676 (2020) 19

15. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4401–4410 (2019) 1, 3, 7, 9, 11, 21
16. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of StyleGAN. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8110–8119 (2020) 1, 3, 7, 19
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proc. ICLR (2015) 19
18. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017) 4, 6, 8, 9, 10, 11, 14, 20, 21, 22, 23
19. Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2661–2671 (2019) 3
20. Li, Y., Zhang, R., Lu, J.C., Shechtman, E.: Few-shot image generation with elastic weight consolidation. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 15885–15896. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/b6d767d2f8ed5d21a44b0e5886680cb9-Paper.pdf 6
21. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017) 4
22. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S.: Least squares generative adversarial networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2794–2802 (2017) 3
23. Mescheder, L., Geiger, A., Nowozin, S.: Which training methods for GANs do actually converge? In: International conference on machine learning. pp. 3481–3490. PMLR (2018) 3, 19
24. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al.: Mixed precision training. arXiv preprint arXiv:1710.03740 (2017) 19
25. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014) 4
26. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018) 3
27. Miyato, T., Koyama, M.: cGANs with projection discriminator. arXiv preprint arXiv:1802.05637 (2018) 19
28. Mo, S., Cho, M., Shin, J.: Freeze discriminator: A simple baseline for fine-tuning GANs. arXiv preprint arXiv:2002.10964 (2020) 9, 10, 11, 14, 20, 21, 22, 23
29. Nguyen, C., Hassner, T., Seeger, M., Archambeau, C.: Leep: A new measure to evaluate transferability of learned representations. In: International Conference on Machine Learning. pp. 7294–7305. PMLR (2020) 3
30. Noguchi, A., Harada, T.: Image generation from small datasets via batch statistics adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2750–2758 (2019) 3
31. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1717–1724 (2014) 3

32. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on knowledge and data engineering **22**(10), 1345–1359 (2009) 3
33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703 (2019) 19
34. Pidhorskyi, S., Adjeroh, D.A., Doretto, G.: Adversarial latent autoencoders. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14104–14113 (2020) 2
35. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020 (2021) 2
36. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. arXiv preprint arXiv:2102.12092 (2021) 2
37. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. arXiv preprint arXiv:1606.03498 (2016) 3
38. Seff, A., Beatson, A., Suo, D., Liu, H.: Continual learning in generative adversarial nets. arXiv preprint arXiv:1705.08395 (2017) 4
39. Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of GANs for semantic face editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9243–9252 (2020) 2, 12
40. Shin, H.C., Tenenholtz, N.A., Rogers, J.K., Schwarz, C.G., Senjem, M.L., Gunter, J.L., Andriole, K.P., Michalski, M.: Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In: International workshop on simulation and synthesis in medical imaging. pp. 1–11. Springer (2018) 13
41. Shu, Y., Kou, Z., Cao, Z., Wang, J., Long, M.: Zoo-tuning: Adaptive transfer from a zoo of models. In: International Conference on Machine Learning. pp. 9626–9637. PMLR (2021) 3
42. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/68053af2923e00204c3ca7c6a3150cf7-Paper.pdf 7
43. Tran, A.T., Nguyen, C.V., Hassner, T.: Transferability and hardness of supervised classification tasks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1395–1405 (2019) 3
44. Viazovetskyi, Y., Ivashkin, V., Kashin, E.: StyleGAN2 distillation for feed-forward image manipulation. In: European Conference on Computer Vision. pp. 170–186. Springer (2020) 12
45. Wang, Y., Gonzalez-Garcia, A., Berga, D., Herranz, L., Khan, F.S., Weijer, J.v.d.: Minegan: effective knowledge transfer from GANs to target domains with few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9332–9341 (2020) 2, 3
46. Wang, Y., Wu, C., Herranz, L., van de Weijer, J., Gonzalez-Garcia, A., Raducanu, B.: Transferring GANs: generating images from limited data. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 218–234 (2018) 3, 4, 5, 10, 11, 14, 20, 21, 22, 23

47. Wu, C., Herranz, L., Liu, X., van de Weijer, J., Raducanu, B., et al.: Memory replay GANs: Learning to generate new categories without forgetting. Advances in Neural Information Processing Systems **31**, 5962–5972 (2018) 4
48. Yoon, J., Drumright, L.N., Van Der Schaar, M.: Anonymization through data synthesis using generative adversarial networks (ADS-GAN). IEEE journal of biomedical and health informatics **24**(8), 2378–2388 (2020) 13
49. Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015) 9, 10, 20, 21
50. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International Conference on Machine Learning. pp. 3987–3995. PMLR (2017) 4
51. Zhai, M., Chen, L., Tung, F., He, J., Nawhal, M., Mori, G.: Lifelong GAN: Continual learning for conditional image generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2759–2768 (2019) 4

A Implementation Details

We evaluated our technique and the baselines using the StyleGAN2 architecture [16]. We kept most of the details unchanged, including network architecture, weight demodulation, regularization, exponential moving average of generator weights, R1 regularization [23], mini-batch size of 32 images, and using the Adam optimizer [17] with $\beta_1 = 0, \beta_2 = 0.99$ and $\epsilon = 10^{-8}$.

In order to introduce conditioning to the unconditioned StyleGAN2 architecture we add the following components to the generator and the discriminator:

- **Generator conditioning.** We add a class embedding layer to the mapping network of the generator, s.t. the input to the generator is noise vector z and one-hot class c. The embedded condition is concatenated to the input z. The first fully-connected layer of the mapping network is modified to support this new size.
- **Discriminator conditioning.** We add a mapping network to the discriminator that gets as an input only a class one-hot vector c (with no noise vector z) and calculates a w vector. We then incorporate this w vector to the final discriminator prediction by a projection [27].

If our input models are unconditioned or conditioned with an insufficient number of classes, we can easily introduce/extend the class embedding layer to the input models to the desired size by adding more rows to the embedding matrix and initialize it randomly.

It is important to notice that we do not rely on any of StyleGAN's features in our solution (or in the baselines), so our solution is agnostic to the input GAN architecture.

A.1 Hyperparameters and training configurations

We used the same hyperparameter configurations as in the PyTorch [33] implementation of StyleGAN2-ada [14], while we did not use the adaptive augmentation capabilities. We used a fixed mapping depth of 8 layers during all our experiments. The hyperparameters were chosen by a random search and are available at the source code.

We used a single NVIDIA RTX 2080 GPU per experiment. We incorporated mixed-precision training [24] in order to speed up the training process.

We trained all our experiments until convergence, which takes about 5M training steps because we start from pre-trained models. Each stage of our two-stage approach (model rooting and merging) takes about 2 days on NVIDIA RTX 2080, thus the total training time is about 4 days.

B Additional experiments

In addition to the experiments reported in the main paper we also compared our method on other datasets. Furthermore, we experimented with mixing models of different architectures and mixing models of different quality.

Table 7: Comparison of FID score w.r.t. the union of all the datasets, for several dataset combinations. Cat, horse, church and bedroom datasets are taken from LSUN [49]

Datasets	cat horse	cat bedroom	cat church	FFHQ horse	FFHQ bedroom	FFHQ church	horse church	horse bedroom	church bedroom
From scratch	20.53	22.81	21.01	13.74	14.96	11.83	12.85	16.96	13.33
TransferGAN [46]	16.73	18.7	17.22	14.4	13.88	11.19	12.27	15.22	11.84
EWC [18]	17.46	17.98	16.75	13.27	14.25	11.11	14.51	15.43	11.07
Freeze-D [28]	16.98	18.53	18.04	13.25	13.14	9.74	10.78	15.81	11.81
Our	**16.46**	**16.7**	**15.62**	**11.28**	**11.5**	**9.52**	**10.61**	**13.9**	**9.91**
Upper bound (real data training)	13.17	14.42	13.65	8.52	8.59	6.25	8.65	10.26	7.01

B.1 Additional datasets

We compared our method using additional classes from the LSUN dataset. As can be seen in Table 7, our method outperforms the baselines in all of our experiments. In addition, we tested the effect of using multiple source datasets, as reported in Table 8. As we can see, our method outperforms the baselines even when mixing seven different models.

B.2 Mixing models with different architectures

For most of our experiments we use the StyleGAN2 framework. However, our method can be used to merge models with different architectures. In the first stage (model rooting) after we choose the root model, the remaining models serve only as data-generators, hence can be of any architecture. In the second stage (model merging), all the rooted models that we create are, by design, of the same architecture as the root.

We evaluated our solution (and the baselines) on merging models with different architectures: a StyleGAN2 model trained on LSUN cat and a custom made model that was trained on LSUN dog. The custom made model was created by removing the mapping network from the StyleGAN architecture and replace it with a simple linear embedding layer. Each of the models was chosen as root, thus in one case the merged model is a StyleGAN2, and in the other case, the merged model is a custom one. As shown in Table 9, our mixing approach outperforms the baselines, in terms of FID score, regardless of the architecture of the root model. Again, it does not mean that the root model is meaningless: choosing the StyleGAN2 architecture for the merged model produces superior results, compared to merging that uses the custom architecture.

We have noticed that both of the source models have comparable FID scores, which leads us to the next question: what happens if we mix models of different FID scores.

Table 8: Comparison of FID score w.r.t. the union of all the datasets, for several dataset combinations. Cat, horse, church, car and bedroom datasets are taken from LSUN [49].

Datasets	FFHQ cat	FFHQ cat dog	FFHQ cat dog car	FFHQ cat dog car horse	FFHQ cat dog car horse bedroom	FFHQ cat dog car horse bedroom church
From scratch	19.61	23.22	24.88	20.56	18.36	18.34
TransferGAN [46]	18.63	20.64	19.34	18.4	17.49	17.29
EWC [18]	19.45	19.47	19.14	18.14	18.56	17.18
Freeze-D [28]	18.17	19.71	19.41	17.8	17.24	17.5
Our	**16.44**	**18.98**	**18.44**	**17.35**	**17.04**	**16.41**
Upper bound (real data training)	11.86	15.93	16.45	16.19	16.88	16.33

B.3 Mixing models of different quality

In order to isolate and identify changes that result in consistent improvements across our various experiments, we mainly focus on comparing models of the same quality: models of the same capacity that were trained on roughly the same dataset size. This raises the question of whether our method is beneficial in the cases where the models are of different quality.

To test under such conditions, we trained a StyleGAN2 model on a reduced version of LSUN dog with an order of magnitude fewer training samples: we evaluated the mixing between a model that was trained on 100K samples of LSUN cat and a model that was trained on 10K samples of LSUN dog. Table 10 demonstrates that in this scenario, our method still outperforms the baselines regardless of the choice of the root model. It is also important to notice that EWC performs significantly worse when the root model is the one that was trained on the smaller dataset, because the inductive bias towards the weights of this weaker model is a bad prior. The other baselines, as well as our method, are much less sensitive. Nevertheless, we can see that choosing the root model to be the model that was trained on the larger dataset yields better results.

C Datasets

We used FFHQ [15] and LSUN [49] datasets for our experiments. We used the entire FFHQ dataset which contains 70K images that are automatically aligned and cropped.

Images in the FFHQ dataset are licensed under either Creative Commons BY 2.0, Creative Commons BY-NC 2.0, Public Domain Mark 1.0, Public Domain

Table 9: **Mixing models of different architectures.** Our method outperforms the baselines, in terms of FID score, regardless of the architecture of the chosen root model

	LSUN cat + LSUN dog	
Root model	LSUN cat (StyleGAN)	LSUN dog (Custom)
Scratch	23.34	23.34
TransferGAN [46]	19.76	22.39
EWC [18]	21.57	21.79
Freeze-D [28]	19.70	21.44
Our	**19.42**	**21.28**

Table 10: **Mixing models of different dataset sizes.** Our method outperforms the baselines, in terms of FID score, regardless of the model that is chosen as a root model. In addition, we can see that EWC performs poorly when initialized with the weaker model

	LSUN cat + LSUN dog	
Root model **# Training samples**	LSUN cat (100K)	LSUN dog (10K)
Scratch	37.46	37.46
TransferGAN [46]	32.52	34.93
EWC [18]	32.18	45.10
Freeze-D [28]	32.03	35.30
Our	**31.71**	**32.59**

CC0 1.0, or U.S. Government Works license. All of these licenses allow free use, redistribution, and adaptation for non-commercial purposes.

The LSUN dataset contains around one million labeled images for each of 10 scene categories and 20 object categories. We used only some of the categories in the dataset (cat, dog, and car) and used only 100K images per class (in order to keep the balance between the FFHQ and the LSUN classes).

We trained the models once and then used the output of the trained models for our experiments. The dataset was used during our experiments only for calculation of the FID metrics. Note that we did not change the behavior of the training process based on the FID score in any way, because we assume that our method should be applicable without any training data. The multivariate Gaussian statistics of the inception features may not be available during the training for the end-user, hence we cannot use it.

Table 11: FID comparison of merging LSUN cat + LSUN dog when training on a higher resolutions

Datasets resolution	128×128	256×256
From scratch	32.58	28.61
TransferGAN [46]	26.90	23.28
EWC [18]	28.26	27.18
Freeze-D [28]	29.00	23.65
Our	**25.12**	**22.45**
Upper bound (real data training)	18.82	18.88

C.1 FID calculations

The results in the tables in the main paper are calculated over images of size 64×64, for efficiency reasons. To make sure that the same trends hold for higher resolutions, we tested our method on images of sizes 128×128 and 256×256 on the LSUN cat and LSUN dog datasets and achieved similar results, as can be seen in Table 11. Each stage of our two-stage approach (model rooting and merging) takes about 4 days on NVIDIA RTX 2080 for resolution 128×128, and about 7 days on NVIDIA A10 for resolution 256×256; thus, the total training time is about 8 days and 14 days, respectively.

D Training

In Figure 5 we can see the convergence rate of the FID that is calculated on the union of the input datasets LSUN cat and LSUN dog (which are semantically close datasets) during the training process. As we can see, our solution converges more quickly and to a lower FID than the baselines.

In Figure 6 we show the FID score that is calculated per class. As we can see, TransferGAN is suffering from catastrophic forgetting on the cat class (left) that is somewhat mitigated by the EWC but it comes at the expense of increasing the FID score of the dog class (right). In contrast, our method starts from a higher FID score on the cat dataset than TrasferGAN/EWC/Freeze-D methods (because they started from the pretrained cat model which achieves better results), but later on, our method achieves a better result.

In Figure 7 we can see the convergence rate of the total FID score when merging two semantically distant datasets: FFHQ and LSUN cat. We can see that our method converges more quickly and to a lower FID than the baselines. As we can see in Figure 8 again, EWC mitigates the catastrophic forgetting of TransferGAN on the FFHQ class and even achieves a better result on this class than our method. But it achieves the worst result on the second class (even worse than the naïve from scratch approach). So all-in-all it is outperformed by our method as can be seen in Figure 7.

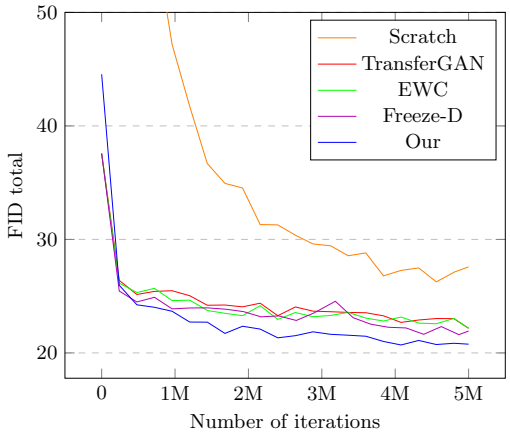


Fig. 5: Convergence rate of the total FID score during the training on LSUN cat + LSUN dog. We can see that our solution achieves the lowest (best) FID score.

E Applications

Semantic editing We demonstrate additional examples for the semantic editing application. We used a merged model of FFHQ and LSUN cat. In Figure 9 we show more examples to Figure 4 in the main paper: we calculated the human pose direction in the $\mathcal{W}$ latent space of the merged generator on images of FFHQ class only by fitting an SVM that separates images with "positive" pose and images with "negative" pose, then we used the calculated hyperplane normal and applied it to images that were generated from both of the classes. Note how the pose direction also applies to the cats, even though it was calculated using human photos.

In addition, we also experimented with semantic directions whose meaning may be less clear or even undefined for some of the classes. We did not expect these manipulations to work, but wanted to investigate their behavior. In Figure 10 we calculate the direction in the latent space that corresponds to the gender of the subject on the FFHQ class, and apply this direction to images for both classes. As can be seen, this direction has a clear effect on the FFHQ class, but not on the LSUN cat class, where it mainly affects the size of the cat. Another example can be seen in Figure 11, where we calculate the "add glasses" direction in the latent space for the FFHQ class. While this direction operates well on the FFHQ class, since the LSUN cat class does not have images of cats with glasses, it is not surprising that the effect is not carried over to cat images. Note, however, that this latent direction does affect the same semantic region — adding glasses is replaced by slightly increasing the cats' eyes.

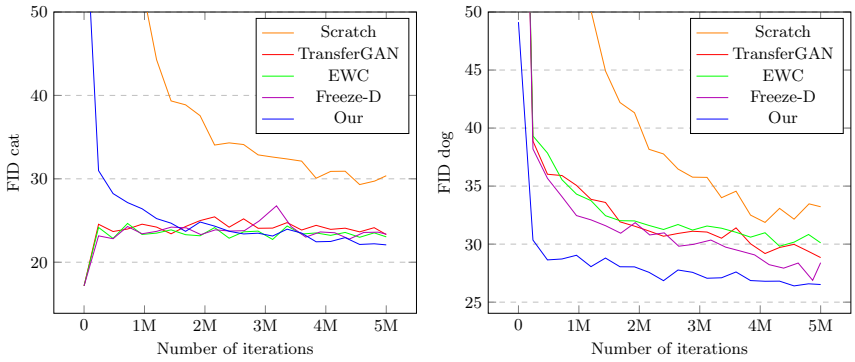


Fig. 6: The FID score that is calculated on the cat class (left) and on the dog class (right). As we can see, the TransferGAN suffers from catastophic forgetting of the cat class (left) that is somewhat mitigated by the EWC but it comes at the expense of increasing the FID score of the dog class (right).

F Uncurated Generation Examples

In Figure 12 and Figure 13 we present uncurated images generated by the input source GAN models, by the baselines, and by our method.

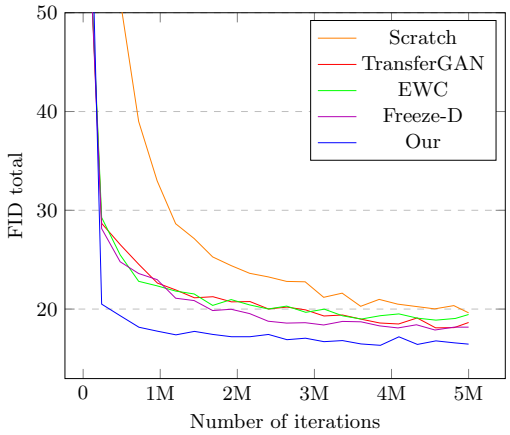


Fig. 7: Convergence rate of the total FID score during training on LSUN cat + FFHQ. Our solution achieves the lowest (best) FID score.

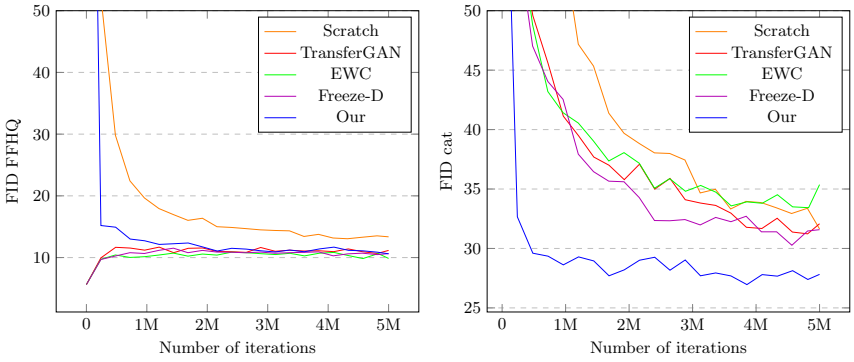
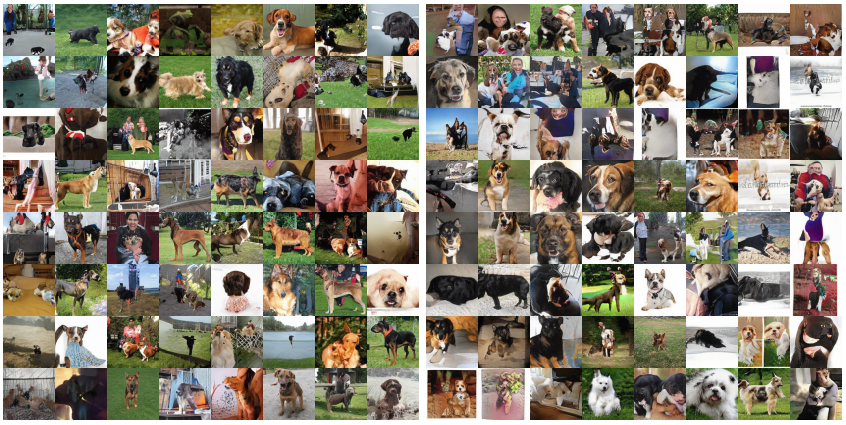


Fig. 8: FID scores calculated separately on FFHQ (left) and on cat (right). TransferGAN suffers from catastophic forgetting of FFHQ (left) that is mitigated by EWC which achieves slightly better results on this dataset than our method, but this comes at the expense of the FID score of the cat class (right), which is the worst for EWC out of all methods.

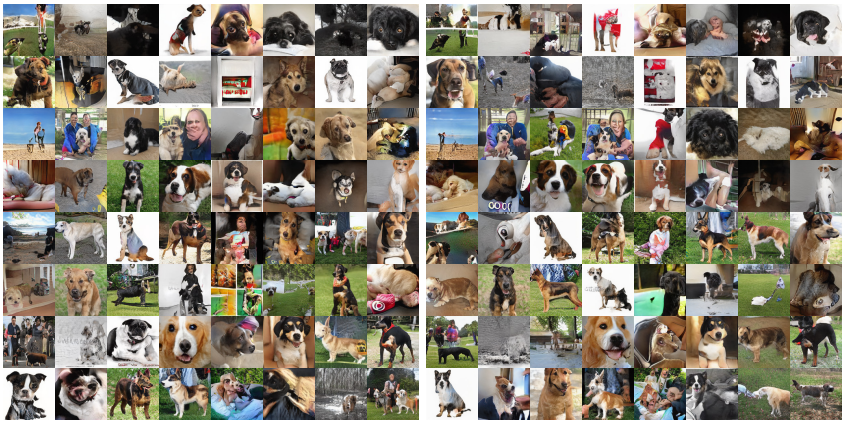
Fig. 9: We determine the pose direction in the latent space of the merged model of FFHQ and LSUN cat using images of the FFHQ class only. We then apply this direction to images from both classes and find that the semantics are largely preserved.

Fig. 10: We determine the gender direction in the latent space of the merged model of FFHQ and LSUN cat using images of the FFHQ class only. We then apply this direction to images from both classes. As expected, this operates accurately only on the FFHQ class.

Fig. 11: We determine the glasses direction in the latent space of the merged model of FFHQ and LSUN cat using images of the FFHQ class only. We then apply this direction to images from both classes. The addition of glasses operates accurately only on the FFHQ class (as expected). On the cat class the same direction enlarges the eyes of the cats.



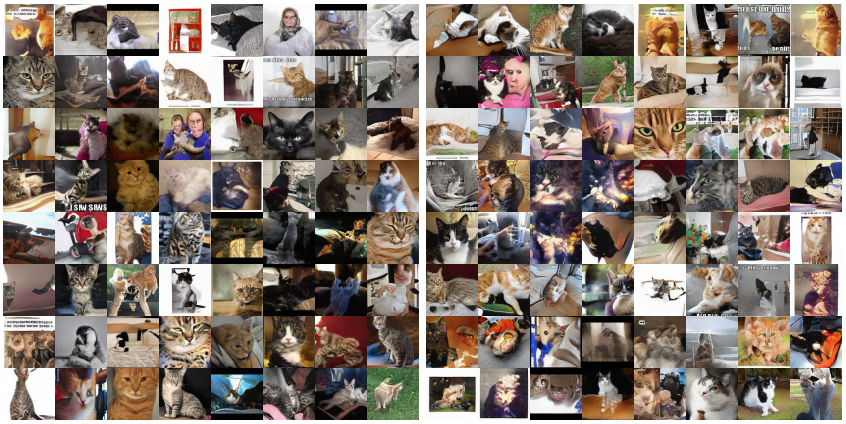
(a) Source model (b) From scratch



(c) TransferGAN (d) FreezeD

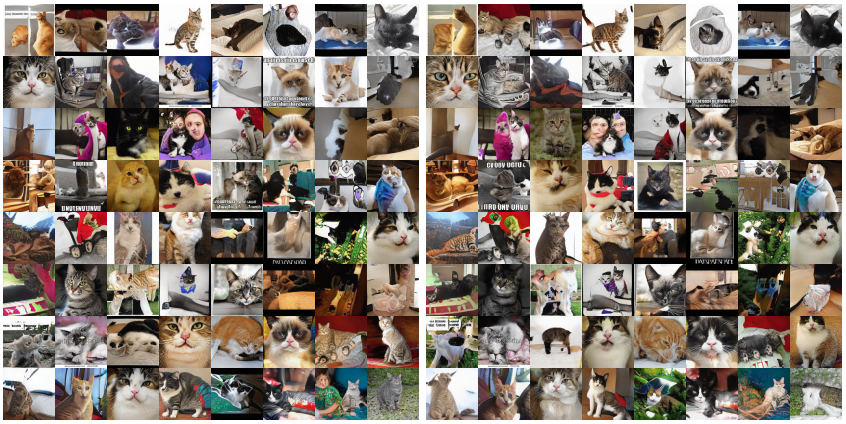
(e) EWC (f) Ours

Fig. 12: Examples of uncurated images that were generated by the source model (a), the baselines (b-e), and our method (f) on LSUN dog dataset.



(a) Source model (b) From scratch

(c) TransferGAN (d) FreezeD



(e) EWC (f) Ours

Fig. 13: Examples of uncurated images that were generated by the source model (a), the baselines (b-e), and our method (f) on LSUN cat dataset.