Diffusion Soup: Model Merging for Text-to-Image Diffusion Models

Benjamin Biggs*[1], Arjun Seshadri*[1], Yang Zou[2], Achin Jain[2], Aditya Golatkar[1], Yusheng Xie[2], Alessandro Achille[1], Ashwin Swaminathan[2], and Stefano Soatto[1]

[1] AWS AI Labs
[2] Amazon AGI Foundations

Abstract. We present Diffusion Soup, a compartmentalization method for Text-to-Image Generation that averages the weights of diffusion models trained on sharded data. By construction, our approach enables training-free continual learning and unlearning with no additional memory or inference costs, since models corresponding to data shards can be added or removed by re-averaging. We show that Diffusion Soup samples from a point in weight space that approximates the geometric mean of the distributions of constituent datasets, which offers anti-memorization guarantees and enables zero-shot style mixing. Empirically, Diffusion Soup outperforms a paragon model trained on the union of all data shards and achieves a 30% improvement in Image Reward (.34 $\rightarrow$ .44) on domain sharded data, and a 59% improvement in IR (.37 $\rightarrow$ .59) on aesthetic data. In both cases, souping also prevails in TIFA score (respectively, 85.5 $\rightarrow$ 86.5 and 85.6 $\rightarrow$ 86.8). We demonstrate robust unlearning—removing any individual domain shard only lowers performance by 1% in IR (.45 $\rightarrow$ .44)—and validate our theoretical insights on anti-memorization using real data. Finally, we showcase Diffusion Soup's ability to blend the distinct styles of models finetuned on different shards, resulting in the zero-shot generation of hybrid styles.

1 Introduction

On paper, the ultimate goal of a foundational image generation model is to approximate, given a large dataset $\mathcal{D}$, the underlying data distribution $p(x)$ generating those images. This view, however, hides the important subtleties that come with real-world data: any real large scale training dataset $\mathcal{D}$ is not just an identically distributed collection of images, but rather a variable entity where new data, coming from different sources covering different domains and usage rights, is frequently added or removed. In these situations, training a monolithic model on all data is problematic. If the data changes, the whole model has to be retrained (at a large cost) to either add new information (continual learning), or remove information that cannot be used anymore (machine unlearning). Moreover, while a single model trained on all the data together can have impressive

* Equal contribution, alphabetical order. Correspondence to: benbiggs@amazon.com.

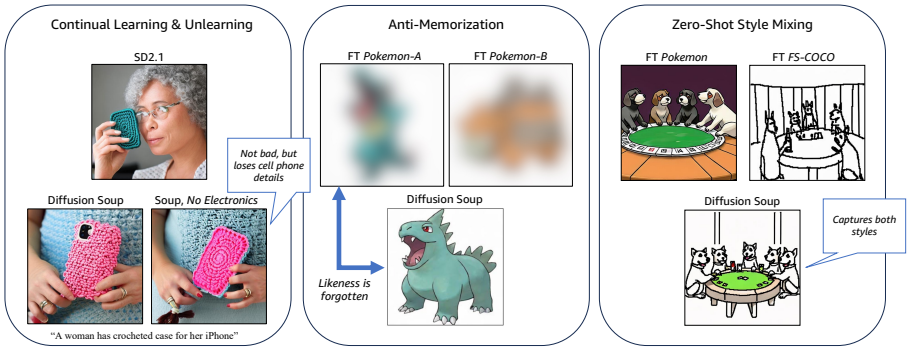

Fig. 1: Diffusion Soup Enables Three Distinct Applications. (1) *Continual Learning & Unlearning*: models trained on various data shards can be added to improve performance or subtracted when removal is necessary. (2) *Zero-Shot Style Mixing*: souping blends the styles into a hybrid of its components with no extra supervision. (3) *Anti-Memorization*: Diffusion Soup prevents memorization while capturing its high level style (note that we blur depictions of inputs in this subfigure).

coverage and overall performance, it often underperforms expert models trained on particular domains (see Table 1), to the detriment of downstream users interested in those particular use-cases.

This dilemma has prompted interest in *compartmentalization* or *mixture of expert* (MoE) methods: rather than treating the model as a monolithic blackbox trained on all data, different sets of parameters are trained on different shards of the training set (corresponding, for example, to different domains or data provided by different users) and then combined at inference time. Compartmentalized models excel at continual learning, unlearning, and anti-memorization tasks. However, in the simplest implementation through ensembling of independent models, they significantly increase inference time — especially as the number of models grows in the hundreds/thousands as would be desirable to have fine control over data provenance. Moreover, ensemble models are significantly more complex to deploy, as they require custom architectures and proper load distribution across instances. The natural question is whether there is a simple method to merge information between different models trained on different subsets of data into a single model, with little or no additional computational overhead.

To answer this question, we introduce *Diffusion Soup*. Diffusion Soup trains separate models on different subsets of data, and then simply averages their weights to obtain a single model. Averaging weights may seem ill-advised, since weights do not live in a linear (vector) space and therefore averaged weights may be far from, and perform worse than, any of its components. However, we show that, when done properly, this strategy not only leads to viable models, but actually outperforms a paragon monolithic model trained on the combination of

Fig. 2: Images from Diffusion Soup Beat SD2.1 and a Combined Paragon. We visualize images generated by averaging the weights of various models finetuned on data shards spanning different categories (*Souped*), and compare them to images from the pretrained model (*SD2.1*) and a paragon model trained all the shards (*Combined*). These images highlight Diffusion Soup's dominance in metrics (See Table 1) for Text-Image Alignment, Aesthetics, and Fidelity. Best viewed in color and zoomed in.

all data. See Fig. 2 for a visualization of results from Diffusion Soup alongside a base model and the paragon.

Our result leverages the insight that the training process *enforces* linearity by means of optimization with gradient descent: Each weight along the training path is modified by an *additive* (linear) increment in a randomized direction. Exploiting the structure of the update, we can show via Taylor expansion that a Diffusion Soup of models finetuned from a shared pretrained checkpoint is expected to (approximately) sample images from the geometric mean of the individual model distributions. In contrast, training a monolithic model samples from their arithmetic mean. Sampling from the geometric mean acts as an implicit regularizer, which counters the decrease in quality due to the overfitting commonly observed during fine-tuning of diffusion models. This justifies our counter-intuitive results that a model soup performs better than a monolithic expert trained with access to all the data.[3]

Empirically, we find that Diffusion Soup outperforms the comparable paragon model trained on all the data shards in two settings: (1) when data shards are grouped by domain (e.g. *Animals*, *Electronics*, etc.), Diffusion Soup excels in TIFA score (85.5 $\rightarrow$ 86.5) and Image Reward (0.34 $\rightarrow$ 0.44); and (2) when data shards are all geared towards Aesthetics, Diffusion Soup prevails in TIFA score (85.6 $\rightarrow$ 86.8), Image Reward (0.37 $\rightarrow$ 0.59), and CLIP Score (0.261 $\rightarrow$ 0.263).

[3] While our results are valid for local perturbations around a pretrained point, empirically we show that the common pre-trained point can be generic enough that individually trained experts starting from it achieve comparable performance than if they were trained from scratch.

We also show that Diffusion Soup: (1) can be used for training-free unlearning (simply removing weights from the average); removing *any* domain's shard decreases performance by at most 1% in Image Reward (.45 $\rightarrow$ .44); (2) the souped model approximates a Near Access Freeness condition, which provides the souped model with better anti-memorization guarantees; (3) can be modified by the user to achieve desired zero-shot *blending* between styles.

2 Related Work

Diffusion models. Diffusion models [12, 21, 40, 42, 45] are state-of-the-art models used for text-to-image generation. These models add Gaussian noise to an image (or its latent representation) in the forward process, and learn the score function to denoise the image in the backward process (introduced by [29]) during generation. [40, 41] condition diffusion models on text to enable text-to-image generation at inference using a technique called classifier-free guidance [22]. [46, 47] show that the forward-backward process obeys a stochastic differential equation, we use this framework in this paper. *Compositional generation.* [4, 7, 14, 18, 20, 30, 57] propose performing compositional generation by merging output flows of diffusion models, methods that often incur increased compute cost. [54] proposes a mixture of experts method by replacing the standard convolutional layers with mixture-of-expert layers which reduce the inference cost and the number of parameters during inference. We show that our method further reduces the inference cost and parameters, and incurs no additional inference or memory overhead beyond that of a single model. *Model merging.* Weight averaging [9, 16, 25, 34, 52] is a popular technique used to improve the accuracy of discriminative models. [31, 32] further extended these methods for linearized convolutional and transformer based models. [3, 37] both proposed and studied model merging methods for generative adversarial networks, but did not consider diffusion models. While the AI art community [1] has long utilized weight averaging to generate high quality images, to the best of our knowledge, we are the first work to quantitatively demonstrate that weight averaging improves text-to-image alignment, reduces memorization at inference, and mixes stylistic components of disparate domains.

3 Preliminaries

Latent diffusion models like Stable Diffusion [41] aim to model a data distribution $p(x_0)$ of images $x_0 \in \mathcal{X}$ which can be sampled at inference time. We use the stochastic differential equations (SDE) based diffusion formalism of [26, 47] to define the basics of diffusion models. Given the initial distribution $p(x_0)$ at time $t = 0$, SDE based formalism transforms it into a reference distribution $p(x_1) = \mathcal{N}(0, I)$ (Gaussian distribution) at time $t = 1$ in the forward process:

$$dx_t = -\frac{1}{2}\beta_t x_t dt + \sqrt{\beta_t} d\rho_t \tag{1}$$

where x_t is the diffused latents at t, $d\rho_t$ is the standard Wiener process, and β_t are time varying diffusion coefficients used to manipulate the signal-to-noise ratio of the diffusion process. The intermediate latents x_t are distributed according to a conditional gaussian distribution $p(x_t|x_0) = \mathcal{N}(x_t; \gamma_t x_0, \sigma_t^2 I)$ by construction where $\gamma_t = \exp(-\frac{1}{2}\int_0^t \beta_t dt)$ and $\sigma_t^2 = 1 - \gamma_t^2$, providing the following marginal: $p(x_t) = \int_{x_0} p(x_t|x_0)dx_0$. [29, 47] shows that the forward process in Eq. (1) can be effectively reversed by the backward diffusion process given by:

$$dx_t = -\frac{1}{2}\beta_t x_t dt + \sqrt{\beta_t} d\rho_t - \nabla_{x_t} \log p(x_t) dt \tag{2}$$

where dt is a decrement in time corresponding to the backward process, introduced by [29]. Eq. (2) reduces sampling from $p(x_0)$ to iterative application of $\nabla_{x_t} \log p(x_t)$ (score function) given an initial noisy samples $x_1 \sim \mathcal{N}(0, I)$. The score function $\nabla_{x_t} \log p(x_t)$ is generally difficult to estimate in practice and is instead learned using a neural network $\epsilon_w(x_t, t)$ and a training dataset D through score-matching [13, 24, 46, 47]. To enhance the usability and control of diffusion models they are often conditioned with textual prompts y to model the conditional distribution $p(x_0|y)$ which modifies the score-estimating neural network as $\epsilon_w(x_t, t, y)$, to now accept y as input during training and inference. Incorporating all these subtleties results in the following optimization problem to train text-to-image diffusion models:

$$\min_w \mathbb{E}_{(x_0,y)\sim p(x_0,y)} \mathbb{E}_t[\|\epsilon_w(x_t, t, y) + \nabla_{x_t} \log p(x_t|x_0)\|] \tag{3}$$

We can obtain an equivalent optimization problem by formulating the forward process in Eq. (1) as a Markov chain and optimizing a variant of the evidence lower bound (ELBO) [21, 45].

4 Diffusion Soup

While Diffusion models are pre-trained on large monolithic datasets (e.g., LAION-5B), in downstream applications it is often more common to have datasets D_i come from different data sources, which may correspond to different data providers, or different domains. We use $\{p^{(i)}(x_0)\}_i$ to represent the collection of distributions corresponding to datasets, $\{D_i\}_{i=1}$. Ensembling the outputs of generative models [4, 18] integrates information from different data sources, but is expensive due to the size of the models. Instead we propose to ensemble the weights of the model, inspired by recent work [2, 17, 27, 32, 33, 51, 56] that shows the fine-tuning dynamics of large models can be approximated with a Taylor series:

$$\epsilon_{w_c + \Delta w_i}(x_t, t, y) \approx \epsilon_{w_c}(x_t, t, y) + \nabla_w \epsilon_{w_c}(x_t, t, y)|_{w=w_c} \Delta w_i \tag{4}$$

where Δw_i is the perturbation of the weights learned during fine-tuning. This result can be leveraged to show that ensembling the outputs corresponds to

ensembling the weights of the models w_i trained on different data sources D_i (sampled from $p^{(i)}(x_0)$). More precisely, we define the souped prediction as:

$$\begin{aligned}\epsilon_{\text{soup}} &\triangleq \underbrace{\epsilon_{\sum_i k_i w_i}}_{\text{Souping}}(x_t,t,y) = \epsilon_{\sum_i k_i(w_c+\Delta w_i)}(x_t,t,y)\\ &\overset{(a)}{\approx} \epsilon_{w_c}(x_t,t,y) + \nabla_w \epsilon_{w_c}(x_t,t,y)|_{w=w_c}\cdot\Big(\sum_i k_i \Delta w_i\Big)\\ &= \sum_i k_i\Big(\epsilon_{w_c}(x_t,t,y) + \nabla_w \epsilon_{w_c}(x_t,t,y)|_{w=w_c}\Delta w_i\Big)\\ &\overset{(b)}{\approx} \sum_i k_i \epsilon_{w_c+\Delta w_i} = \sum_i k_i \epsilon_{w_i}(x_t,t,y) = \epsilon_{\text{ensemble}} \end{aligned} \tag{5}$$

where $k_i > 0$, such that $\sum_i k_i = 1$ is a hyper-parameter which can be tuned by the user, and $(a), (b)$ follow from using the first order Taylor series approximation of the network. To summarize, Diffusion Soup approximates ensembling, and involves fine-tuning $n-$diffusion models ($\{\epsilon_{w_i}(x_t,t,y)\}_i$) on $n-$data sources ($\{p^{(i)}(x_0)\}_i$) respectively, and averaging the parameters.

4.1 Sampling Distribution for Souping

Modifying the score (or the $\epsilon(x_t)$) during the backward diffusion process changes the distribution which is sampled by the model. For instance, using Eq. (2) samples images from the distribution $p(x_0)$. In our case, since we modify the individual $\{\epsilon_{w_i}(x_t,t,y)\}$ to the soup $\epsilon_{\sum_i k_i w_i}(x_t,t,y)$ it is essential to identify the sampling distribution of the model. Depending on the choice of k_i we can show that the souped model can either sample from various aggregations of the set of distributions $\{p^{(i)}(x_0)\}$. This is described in the following result:

Proposition 1. *(Geometric Mean) Let $\nabla_{x_t}\log p^{(i)}(x_t)$ be the marginal score in Eq. (2) where $x_t = \gamma_t x_0 + \sigma_t \epsilon$, with $x_0 \sim p^{(i)}(x_0)$ and $\epsilon \sim \mathcal{N}(0,I)$. Let $\epsilon_{w_i}(x_t,t,y)$ be a neural network with sufficient capacity trained to match $\nabla_{x_t}\log p^{(i)}(x_t)$. Under Eq. (2)'s conditions on the sampling procedure, $(1/n)\sum_{i=1}^n \nabla_{x_t}\log p^{(i)}(x_t)$ generates samples from $Z^{-1}\big(\prod_i p^{(i)}(x_0)\big)^{1/n}$. Furthermore, using Eq. (4), ϵ_{soup} with $k_i = 1/n$ generates samples from $Z^{-1}\big(\prod_i p^{(i)}(x_0)\big)^{1/n}$.*

The previous result states Diffusion Soup with $k_i = 1/n$ generates samples from the geometric mean of the distribution of the individual models, which can be useful to provide memorization-free generation [20,50]. Conversely, a proposition in the appendix shows that souping, with an appropriately chosen value of k_i, can theoretically sample from the arithmetic mean, representing the union of all data (equivalent to training a model on the union). In this work, k_i is fixed across prompts and timesteps, and set to be uniform ($k_i = 1/n$) or chosen using various greedy approaches, as we detail below.

4.2 Greedy Souping

In Eq. (5) we show that the optimal weights after souping are a linear combination of the individual weights w_i (data sources/experts), $w_{\text{soup}} = \sum_i k_i w_i$, where we treat k_i as hyper-parameters to be optimized before inference subject to the constraint $\sum_i k_i = 1$. Let $L(w_{\text{soup}}, D_{\text{val}})$ be a evaluation metric used assess the quality of the diffusion model, for example, TIFA score or Image Reward. The following optimization problem solves for the optimal coefficients k_i:

$$\{k_i^*\}_i = \underset{\sum_i k_i = 1, k_i > 0}{\operatorname{argmin}} L\big(\textstyle\sum_i k_i w_i, D_{\text{val}}\big) \tag{6}$$

Obtaining a closed form expression for Eq. (6) is intractable for diffusion model evaluation, and so we consider two greedy approaches to obtain coefficients. The first, Greedy Soup begins with the best performing individual weight and incrementally soups weights in order of individual performance—keeping the weights if they improve performance and discarding them if they do not. The second, Reverse Greedy Soup begins with the uniform soup and removes weights in order of increasing performance.

5 Analysis

5.1 Near Access Freeness of Diffusion Soup

Diffusion models often memorize some of their training data [5, 44] posing a risk of reproducing samples from the training dataset. [50] recently proposed a mathematical framework based on near-access freeness (NAF) to prevent generative models from memorizing samples present in the training data. Let D be the dataset with some samples $C = \{c_i\}_i$, and $D = D_1 \bigcup D_2$ be two disjoint splits of the dataset. Let $p^{(1)}(x|y), p^{(2)}(x|y)$ (where x is the image and y is the caption) be two diffusion models trained on D_1, D_2 respectively. Then Algorithm 3 from [50] (CP$-\Delta$) shows that sampling from the geometric mean $p(x|y) = Z^{-1}\sqrt{p^{(1)}(x|y)p^{(2)}(x|y)}$ provides anti-memorization guarantees with respect to any sample c_i present in only one of D_1 or D_2. That is, it satisfies ε-NAF:

$$\Delta(p(x|y)||\text{safe}_C(x|y)) \leq \varepsilon \tag{7}$$

where $\text{safe}_C(x|y)$ is a model which has not been trained on c_i, Δ is a divergence between two distributions (e.g., Kullback-Leibler divergence). Sampling from $p(x|y) = Z^{-1}\sqrt{p^{(1)}(x|y)p^{(2)}(x|y)}$ is difficult for diffusion models. However, sampling can be performed implicitly in the backward step during score computation [14, 18] as $\nabla \log p(x|y) = (1/2)\big(\nabla \log p^{(1)}(x|y) + \nabla \log p^{(2)}(x|y)\big)$. With this observation we have the following result:

Proposition 2. *(ε-NAF for Diffusion Soup) Let $\epsilon_{soup}(x_t, t, y) \triangleq \epsilon_{\frac{1}{2}(w_1+w_2)}(x_t, t, y)$ be the soup model obtained by training w_i on D_i, such that $D = D_1 \cup D_2$, and $D_1 \cap D_2 = \varnothing$. Then, under certain conditions, sampling using $\epsilon_{soup}(x_t, t, y)$ satisfies ε-NAF for some ε which depends only on D_1 and D_2.*

The above proposition suggests that, for different choices of D_1 and D_2 we can guarantee different subsets $C = \{c_i\} \subset D$ of NAF-protected samples. If all training samples require anti-memorization ($C = D$), then it suffices to pick D_1 and D_2 to be disjoint partitions of D. Conversely, in a mixed privacy setting [19], we have a *public* set D_{publ} set for which anti-memorization is not required, and a *private* set D_{priv} that requires anti-memorization. In this case, we can make better use of the data by picking $D_1 = D_{\text{publ}}$ to be a core safe set used to pretrain the model, and $D_2 = D_{\text{publ}} \cup D_{\text{priv}}$ to be the fine-tuning set. We explore both ideas for Diffusion models in Section 6.2.

5.2 Computational efficiency

Let M denote the memory consumption (i.e. parameters) during inference, T the time complexity of a forward pass. Compartmentalizing n models has a worst case complexity of $\mathcal{O}(nM), \mathcal{O}(nT)$ for the memory and time respectively, while ensembling [7,54,57] methods have $\mathcal{O}(nM), \mathcal{O}(T)$ complexity respectively. Diffusion Soup reduces the memory complexity by averaging the weights into a single model, resulting in an optimal $\mathcal{O}(M), \mathcal{O}(T)$ complexity. Souping reaps the benefits of multiple specialists and data sources without additional overhead.

5.3 Unlearning

Given n different data sources $D_f = \{D_i\}_i$, and their corresponding diffusion model weights w_i, Diffusion Soup provides a single set of weights $w_{\text{soup}} = \sum_i k_i w_i$. Just as adding new data sources to the soup is straightforward, removal is just as easy: should a data source D_i decides to withdraw their data, we can simply update the souped weights as, $w_{\text{soup}} := \frac{w_{\text{soup}} - k_i w_i}{1 - k_i}$. Should only a subset of data from D_i be removed, we can simply reset the diffusion model ϵ_{w_i} to weights w_c, and fine-tune it with the remaining data in D_i. Souping thus enables efficient disgorgement of data.

6 Experiments

We empirically evaluate Diffusion Soup's ability to achieve training free continual learning and unlearning, provide anti-memorization guarantees, and blend disparate styles into unique hybrids. Our experiments fall into five broad application domains: (1) specialist aggregation, (2) aesthetic enhancement, (3) unlearning (4) blending artistic styles, and (5) anti-memorization. We use Stable Diffusion 2.1 (SD2.1) as the pretrained model, and follow the finetuning procedure proposed in [11]. Our results are not unique to SD2.1, and broadly apply to any Diffusion Model. Details about hyperparameters are in the supplement.

6.1 Datasets and Evaluation Metrics

Datasets We use publicly-available datasets for our experiments. (1) *Souping Specialists:* We generate category-specific datasets and finetune on them to obtain category specialists. These datasets are constructed by first using CLIP [39]

Table 1: Souping a collection of specialist models produces a high-quality generalist. We consider data shards that allow the model to specialize in a range of image domains. Souping these specialist together outperforms all individual models. See Figure 1 for a visualization of Diffusion Soup's performance.

		TIFA	IR	CLIP
Base	SD2.1	85.5	0.30	0.259
SFT	Animals (AN)	86.0	0.41	0.256
	Body Parts (BP)	85.6	0.28	0.257
	Electronics (EL)	85.9	0.42	**0.260**
	Accessories (AC)	85.3	0.38	0.257
	Clothes (CL)	85.0	0.35	0.256
	Food (FO)	85.7	0.37	0.259
	Hardlines (HA)	85.6	0.35	**0.260**
	Products (PR)	85.3	0.37	0.258
	Vehicles (VE)	85.2	0.33	0.257
Combined	All	85.5	0.34	0.257
Souped	Uniform	86.2	**0.45**	**0.260**
	Reverse Greedy	86.3	0.43	**0.260**
	Greedy	**86.5**	0.44	0.259

to retrieve the top 10K highest scoring images for each category, and then using the LAION aesthetic score [43] predictor to retain images with an aesthetic score above 6, leaving approximately a thousand images per category. (3) *Style Mixing:* We use the Pokemon dataset [36] which contains images of Pokemon characters along with BLIP [28] captions, and the FS-COCO dataset [10] to generate fine-tuned checkpoints. (4) *Anti-Memorization and Unlearning:* We use the Pokemon dataset to evaluate anti-memorization and MSCOCO for unlearning.

Evaluation Metrics We consider three metrics commonly used in benchmarking the performance of diffusion models: (1) *Text-to-Image Faithfulness evaluation with Question Answering* [23] (TIFA Score) measures the faithfulness of a generated image to its text input via visual question answering (VQA), which we apply to 2k collected prompts from the MSCOCO Dataset. (2) *Image Reward* [53] (IR) is a scoring model that captures human preferences on image-text alignment, fidelity, and harmlessness. (3) *CLIP Score* [39] (CLIP) captures text-image alignment via the cosine similarity between CLIP embeddings for the generated image and the text prompt.

6.2 Applications

Continual Learning from Multiple Data Subsets The different data subsets that compartmentalization methods operate on are not chosen in practice but rather imposed by the changing nature of data. We therefore showcase the performance of Diffusion Soup on two broad groupings that data subsets often fall into: different *domains* of data with different usage rights, or *task-specific* data provided by different users.

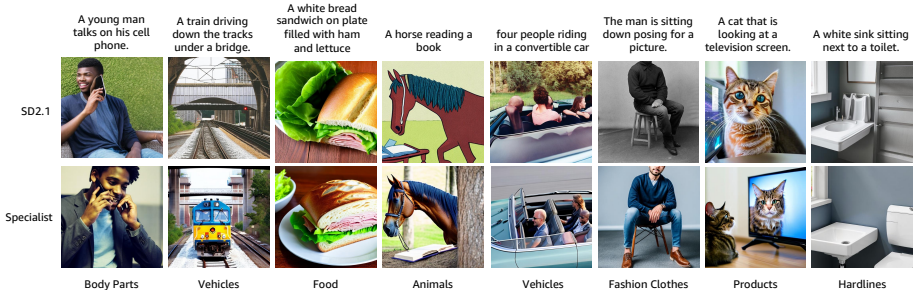

Fig. 3: Finetuning Models on Category Specific Data Subsets Produces Specialists. We visualize results from finetuning SD2.1 models on various data subsets shards by category. The finetuning process specializes the diffusion model to the subset's category: for example, the Body Parts dataset enhances the prominence of fingers, and the Fashion Clothes dataset enhances outfits. These models can be souped together to obtain generalized models that outperform all specialists (See Table 1).

Soup of Specialists. To represent different domains of data, we construct shards corresponding to diverse categories such as *Animals*, *Food*, *Fashion Clothes*, etc and finetune SD2.1 to obtain domain specialists corresponding to each shard. Fig. 3 visualizes each specialist's ability to improve features corresponding to their domain over SD2.1, e.g. *Animals* improve animal photorealism. We now compare various souping methods alongside these specialists and the paragon model trained on the union of all shards, and showcase our findings in Table 1. We find that our finetuned specialists both outperform base SD2.1 in TIFA and IR, and often outperform the combined model paragon, confirming prior work [4,18] demonstrating the value of training specialist models. A uniform soup of the specialist models outperforms all specialists and the paragon in TIFA, IR, and CLIP Score, indicating superior image quality and alignment compared to all base and individual specialist models. Our two greedy souping approaches further optimize TIFA and achieve the highest performing TIFA scores of 86.3 and 86.5. These image alignment optimizations come at a slight cost in image quality however, as indicated by the slight decrease in IR. Regardless, the approaches outperform all specialists and the combined model paragon in TIFA and IR, resulting in a Pareto optimal curve of souping options to choose from.

We visualize the improvements of Diffusion Soup over the base model and the combined model paragon in Fig. 1, highlighting better image alignment, fewer artifacts, and improved aesthetics. Unlike other approaches such as ensembling or MoE methods, these improvements come without compromising on memory consumption and runtime, a significant advance for compartmentalization.

Enhancing Image Aesthetics. We next consider data shards all pertaining to a single task, aesthetic enhancement. We construct 5 highly aesthetic subsets of MSCOCO, AE1-5, ordered by decreasing aesthetic quality to simulate the diversity among data shards stemming from different providers. Once again, we

Table 2: Diffusion Soup Outperforms Base and Combined Models in Aesthetic Enhancement. We consider data shards that all have one task in common, aesthetic enhancement. Our souped results outperform all base models and the combined model in which a single model is trained on a union of the datasets.

	Base & SFT			Combined (Cumulative)			Souped (Cumulative)		
	TIFA	IR	CLIP	TIFA	IR	CLIP	TIFA	IR	CLIP
SD2.1	85.5	0.30	0.259	-	-	-	-	-	-
AE1	86.5	0.53	**0.263**	86.5	0.53	**0.263**	86.5	0.53	**0.263**
AE2	86.3	0.54	0.261	86.0	0.40	0.261	**86.7**	**0.60**	**0.263**
AE3	86.1	0.47	0.262	85.7	0.38	0.261	**86.8**	**0.60**	**0.263**
AE4	86.2	0.43	0.260	86.0	0.39	0.261	**86.8**	**0.59**	**0.263**
AE5	85.9	0.45	0.262	85.6	0.37	0.261	**86.8**	**0.59**	**0.263**

compare various souping approaches to models trained on each shard, as well as a paragon trained on the union. To shed further light into the gap between the Diffusion Soup and the Paragon, we show *cumulative* results for both as we take each aesthetic subset into consideration.

Our results are highlighted in Table 2. We find that all finetuned models outperform base SD2.1 in TIFA and IR, but especially on IR, indicative of the aesthetic quality improvements that IR is particularly sensitive to. Once again, we find that a uniform Diffusion Soup outperforms base SD2.1, all finetuned models, and the combined paragon model on TIFA, IR and CLIP, respectively obtaining scores of 86.8, .59, and .263. The dominance of Diffusion Soup is also shown cumulatively, with the method outperforming the combined paragon model with every additional aesthetic dataset. As continual learning progresses over a growing number of dataset shards, Diffusion Soup continues to improve in performance and increase its lead on a combined model paragon. Curiously, we find that some individual models outperform the combined model paragon – while these models are not specialists in the sense of the previous section, they appear to still specialize in the particular types of aesthetic enhancements present in each data shard, some of which better optimize for TIFA and IR than a combined dataset.

Machine Unlearning We next consider settings where data must be removed, or cannot be used anymore. For Diffusion Soup, just as adding data shards consisted simply of averaging models trained on those shards, removing shards can be trivially implemented through weighted subtraction. We demonstrate unlearning using our category datasets from which domain specialists are constructed, demonstrating that domain specialists can be removed from our model soup without meaningfully compromising the performance of the soup. Remarkably, we find that removing *any* specialist reduces the performance of the soup by at most 1% in IR. Fig. 4 displays our full results, and shows that leaving any one shard out still produces a model soup that closely mirrors the performance of uniform soup in all metrics, and continues to outperform SD2.1 even in the worst case scenario, the removal of the *Clothes* model. In some cases the

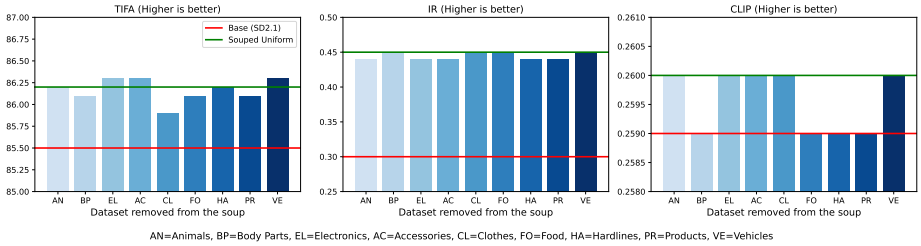

Fig. 4: Removing Any Individual Data Shard Does Not Meaningfully Reduce Performance. We soup specialists leaving one out at a time to demonstrate that no individual specialist significantly affects the quality of the generalist. Our results show that Diffusion Soup can be used for Machine Unlearning. From left-to-right, graphs show TIFA, Image Reward and CLIP Score. Performance of uniform soup model is in *green*, and SD2.1 in *red*.

removal of models improves the TIFA score, and this is the principle exploited by the Reverse Greedy Soup. However, in practice one cannot select the data that is removed, and we view our Fig. 4 as supporting a uniform performance guarantee. The cumulative souping results of Table 2 when read from bottom up serves as yet another demonstration of robust unlearning; indeed a model soup removing data shards AE4 and AE5 performs identically in TIFA score to the uniform soup over all shards, and further removal only results in a mild reduction in performance.

Zero-Shot Style Mixing We employ Diffusion Soup to merge the distinct styles of two separate datasets— FS-COCO's sketch-like imagery and Pokemon illustrations—in a zero-shot manner, i.e. without the need for hybrid training samples. Fig. 5 displays the results of this style fusion. The first row highlights the model tuned on the Pokemon dataset and the second row showcases the FS-COCO-tuned model which generates monochrome line-art sketches. The third row demonstrates that that souping these two models yields generation of a new hybrid style of line-art cartoon characters, without using any hybrid training examples. Our findings are the first to link model souping to Style Mixing, and stem from our novel application of souping to Diffusion Models, and more broadly generative models. We hypothesize that Style Mixing derives from Diffusion Soup sampling from the Geometric Mean of the constituent distributions, formalized in Proposition 1, and therefore distinguishes our results from the outputs produced by training a model on the union of the two datasets. We leave further exploration of this hypothesis for future work.

Anti-Memorization We demonstrate that souping reduces memorization in Diffusion Models with two distinct scenarios, and display our results in Fig. 6. In the first scenario, we randomly split the Pokemon dataset into two shards

Fig. 5: Diffusion Soup merges finetuned models to create hybrid styles. We apply Diffusion Soup to models finetuned on Pokemon (*Row 1*) and FS-COCO (*Row 2*) to create a hybrid style (*Row 3*). The results are zero-shot since we do not have examples of the hybrid style for training.

S1 and S2, and train separate models on each shard. Even though both models reproduce *some* Pokemon, their soup surprisingly does not. We visualize an example in the bottom-middle of Fig. 6, where Pokemon-Soup avoids reproducing samples from the Pokemon dataset while capturing its style. We formalize this finding at a distribution level using CLIP scores. Taking each prompt from the Original Pokemon Dataset, we measure the distance between the Original images and model generated images using CLIP scores. These values together chart a *distance distribution* of model outputs to the original dataset, and we visualize such distributions in the top-left of Fig. 6. There, the two models trained on shards (shown in *blue* and *orange*) each reproduce some samples from the original dataset, as evidenced by low CLIP score peaks of the distributions. The souped model's distance distribution (in *green*) however is shifted considerably to the right, demonstrating that it generates images which are far from the Original data, i.e. avoids memorizing the dataset.

In the second scenario, we soup a model trained on the complete Pokemon dataset with a model trained on a completely disjoint domain—the *Animals* model from Table 1. We visualize an example in the bottom-left of Fig. 6, where Pokemon-*Animals*-Soup once again avoids memorization while capturing style. We plot distance distributions in the top-middle of Fig. 6, where the Pokemon model (*blue*) reproduces the Pokemon data, and the *Animals* model (*orange*) is unsurprisingly far away from it. The souped model's distribution (*green*) again sits to the right of that of the Pokemon model, indicative of its reduced memorization. To compare the effect of the two souping approaches, we compare Pokemon-Soup and Pokemon-*Animals* Soup's distance distributions in the top right of Fig. 6, and find that the latter reduce memorization more than the

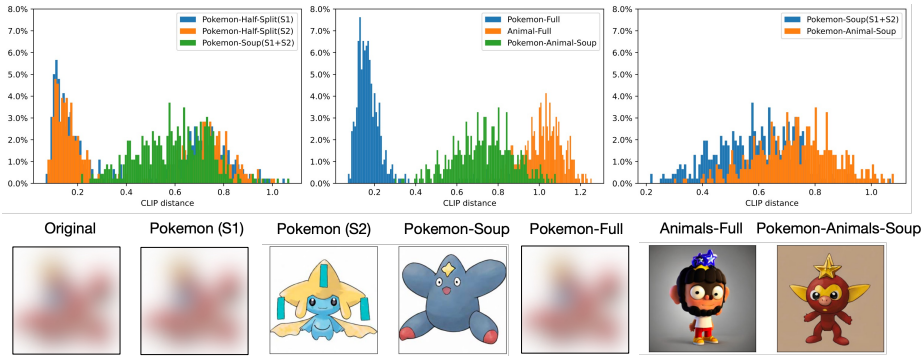

Fig. 6: Diffusion Soup reduces dataset memorization. (Bottom) Models (e.g. Pokemon-Full) trained on Pokemon data (Original) tend to reproduce some training samples at inference time. Note that we blur depictions of inputs in this subfigure. However, souping models (Pokemon-Soup) trained on disjoint subsets of Pokemon (S1 and S2) significantly reduces the memorization when compared to the Pokemon-Full. The same holds for cross-domain souping (Pokemon-*Animals*-Soup), which soups the Pokemon model (Pokemon-Full) with a model trained on a safe dataset (*Animals*-Full). **(Top)** We formalize this anecdotal result with a plot of the CLIP distance of various datasets to the Original Pokemon dataset (*Top Left*). Larger distance or mass towards right implies lower memorization. Souping Pokemon subsets increases the CLIP distance (top middle) and therefore reduces reproduction of Pokemon. Souping Pokemon with *Animals* reduces memorization further (*Top Right*). *Prompt used in figure: "a cartoon character with a star on his head".*

former, but requires an additional dataset and corresponding model. These two methods are those both effective anti-memorization strategies, each with unique tradeoffs for a practitioner to navigate.

Limitations Diffusion Soup performs a variety of tasks with no additional inference cost, yet it still requires training n model shards, and continual training on every new data shard. Also, averaging weights with a particular weighting of each model must be performed globally, and cannot be performed per sample without incurring the inference costs associated with ensembling. Although Adapters can help reduce both the training and reweighting burden, they come at the cost of reduced performance.

7 Conclusion

We present Diffusion Soup, a novel compartmentalization method in the image generation domain. While our contribution reflects the first introduction of model souping to large scale generative AI, its potential to impact performance in other domains is untold. We leave as future work to explore souping in other domains such as language modeling, or visual question answering — domains that can greatly magnify the impact of our work.

Acknowledgements

We thank Tiffany Deng, Hao Li, Pip Liggins, Orchid Majumder, R. Manmatha, Yash Patel, CJ Taylor, Ying Wang, and Maggie White for their helpful comments, pointers, and feedback.

References

1. Stable Diffusion ComfyUI. https://github.com/comfyanonymous/ComfyUI (2023)
2. Achille, A., Golatkar, A., Ravichandran, A., Polito, M., Soatto, S.: Lqf: Linear quadratic fine-tuning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15729–15739 (2021)
3. Avrahami, O., Lischinski, D., Fried, O.: Gan cocktail: mixing gans without dataset access. In: European Conference on Computer Vision. pp. 205–221. Springer (2022)
4. Balaji, Y., Nah, S., Huang, X., Vahdat, A., Song, J., Kreis, K., Aittala, M., Aila, T., Laine, S., Catanzaro, B., et al.: ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. arXiv preprint arXiv:2211.01324 (2022)
5. Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., Balle, B., Ippolito, D., Wallace, E.: Extracting training data from diffusion models. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 5253–5270 (2023)
6. Chen, J., Yu, J., Ge, C., Yao, L., Xie, E., Wu, Y., Wang, Z., Kwok, J., Luo, P., Lu, H., et al.: Pixart-alpha: Fast training of diffusion transformer for photorealistic text-to-image synthesis. ICLR 2024 (2023)
7. Chen, Z., Deng, Y., Wu, Y., Gu, Q., Li, Y.: Towards understanding mixture of experts in deep learning. arXiv preprint arXiv:2208.02813 (2022)
8. Cheng, X., Bartlett, P.: Convergence of langevin mcmc in kl-divergence. In: Algorithmic Learning Theory. pp. 186–211. PMLR (2018)
9. Choshen, L., Venezian, E., Slonim, N., Katz, Y.: Fusing finetuned models for better pretraining. arXiv preprint arXiv:2204.03044 (2022)
10. Chowdhury, P.N., Sain, A., Bhunia, A.K., Xiang, T., Gryaditskaya, Y., Song, Y.Z.: Fs-coco: Towards understanding of freehand sketches of common objects in context. In: ECCV (2022)
11. Dai, X., Hou, J., Ma, C.Y., Tsai, S., Wang, J., Wang, R., Zhang, P., Vandenhende, S., Wang, X., Dubey, A., Yu, M., Kadian, A., Radenovic, F., Mahajan, D., Li, K., Zhao, Y., Petrovic, V., Singh, M.K., Motwani, S., Wen, Y., Song, Y., Sumbaly, R., Ramanathan, V., He, Z., Vajda, P., Parikh, D.: Emu: Enhancing image generation models using photogenic needles in a haystack (2023)
12. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in neural information processing systems **34**, 8780–8794 (2021)
13. Dockhorn, T., Vahdat, A., Kreis, K.: Score-based generative modeling with critically-damped langevin diffusion. arXiv preprint arXiv:2112.07068 (2021)
14. Du, Y., Durkan, C., Strudel, R., Tenenbaum, J.B., Dieleman, S., Fergus, R., Sohl-Dickstein, J., Doucet, A., Grathwohl, W.S.: Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In: International conference on machine learning. pp. 8489–8510. PMLR (2023)
15. Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A.H., Chechik, G., Cohen-Or, D.: An image is worth one word: Personalizing text-to-image generation using textual inversion (2022). https://doi.org/10.48550/ARXIV.2208.01618, https://arxiv.org/abs/2208.01618

16. Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D.P., Wilson, A.G.: Loss surfaces, mode connectivity, and fast ensembling of dnns. Advances in neural information processing systems **31** (2018)
17. Golatkar, A., Achille, A., Ravichandran, A., Polito, M., Soatto, S.: Mixed-privacy forgetting in deep networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 792–801 (2021)
18. Golatkar, A., Achille, A., Swaminathan, A., Soatto, S.: Training data protection with compositional diffusion models. arXiv preprint arXiv:2308.01937 (2023)
19. Golatkar, A., Achille, A., Wang, Y.X., Roth, A., Kearns, M., Soatto, S.: Mixed differential privacy in computer vision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8376–8386 (2022)
20. Golatkar, A., Achille, A., Zancato, L., Wang, Y.X., Swaminathan, A., Soatto, S.: Cpr: Retrieval augmented generation for copyright protection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12374–12384 (2024)
21. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
22. Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)
23. Hu, Y., Liu, B., Kasai, J., Wang, Y., Ostendorf, M., Krishna, R., Smith, N.A.: Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. arXiv preprint arXiv:2303.11897 (2023)
24. Hyvärinen, A., Dayan, P.: Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research **6**(4) (2005)
25. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407 (2018)
26. Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. Advances in Neural Information Processing Systems **35**, 26565–26577 (2022)
27. Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., Pennington, J.: Wide neural networks of any depth evolve as linear models under gradient descent. Advances in neural information processing systems **32** (2019)
28. Li, J., Li, D., Xiong, C., Hoi, S.: Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In: ICML (2022)
29. Lindquist, A., Picci, G.: On the stochastic realization problem. SIAM Journal on Control and Optimization **17**(3), 365–389 (1979)
30. Liu, N., Li, S., Du, Y., Torralba, A., Tenenbaum, J.B.: Compositional visual generation with composable diffusion models. In: European Conference on Computer Vision. pp. 423–439. Springer (2022)
31. Liu, T.Y., Golatkar, A., Soatto, S.: Tangent transformers for composition, privacy and removal. arXiv preprint arXiv:2307.08122 (2023)
32. Liu, T.Y., Soatto, S.: Tangent model composition for ensembling and continual fine-tuning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 18676–18686 (2023)
33. Malladi, S., Wettig, A., Yu, D., Chen, D., Arora, S.: A kernel-based view of language model fine-tuning. In: International Conference on Machine Learning. pp. 23610–23641. PMLR (2023)
34. Matena, M., Raffel, C.: Merging models with fisher-weighted averaging, 2021. arXiv preprint arXiv:2111.09832

35. Neal, R.M.: Annealed importance sampling. Statistics and computing **11**, 125–139 (2001)
36. Pinkney, J.N.M.: Pokemon blip captions. https://huggingface.co/datasets/lambdalabs/pokemon-blip-captions/ (2022)
37. Pinkney, J.N., Adler, D.: Resolution dependent gan interpolation for controllable image synthesis between domains. arXiv preprint arXiv:2010.05334 (2020)
38. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R.: SDXL: Improving latent diffusion models for high-resolution image synthesis. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=di52zR8xgf
39. 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. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
40. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: International Conference on Machine Learning. pp. 8821–8831. PMLR (2021)
41. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
42. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512 (2022)
43. Schuhmann, C.: LAION-Aesthetics. https://github.com/christophschuhmann/improved-aesthetic-predictor (2022)
44. Somepalli, G., Singla, V., Goldblum, M., Geiping, J., Goldstein, T.: Understanding and mitigating copying in diffusion models. Advances in Neural Information Processing Systems **36** (2024)
45. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
46. Song, Y., Durkan, C., Murray, I., Ermon, S.: Maximum likelihood training of score-based diffusion models. Advances in Neural Information Processing Systems **34**, 1415–1428 (2021)
47. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020)
48. Tan, W.R., Chan, C.S., Aguirre, H., Tanaka, K.: Improved artgan for conditional synthesis of natural image and artwork. IEEE Transactions on Image Processing **28**(1), 394–409 (2019). https://doi.org/10.1109/TIP.2018.2866698, https://doi.org/10.1109/TIP.2018.2866698
49. Vempala, S., Wibisono, A.: Rapid convergence of the unadjusted langevin algorithm: Isoperimetry suffices. Advances in neural information processing systems **32** (2019)
50. Vyas, N., Kakade, S., Barak, B.: Provable copyright protection for generative models. arXiv preprint arXiv:2302.10870 (2023)
51. Wei, T., Guo, Z., Chen, Y., He, J.: Ntk-approximating mlp fusion for efficient language model fine-tuning (2023)
52. Wortsman, M., Ilharco, G., Gadre, S.Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A.S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al.: Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In: International Conference on Machine Learning. pp. 23965–23998. PMLR (2022)

53. Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., Dong, Y.: Imagereward: Learning and evaluating human preferences for text-to-image generation. Advances in Neural Information Processing Systems **36** (2024)
54. Xue, Z., Song, G., Guo, Q., Liu, B., Zong, Z., Liu, Y., Luo, P.: Raphael: Text-to-image generation via large mixture of diffusion paths. Advances in Neural Information Processing Systems **36** (2024)
55. Yang, K.Y., Wibisono, A.: Convergence in kl and rényi divergence of the unadjusted langevin algorithm using estimated score. In: NeurIPS 2022 Workshop on Score-Based Methods (2022)
56. Zancato, L., Achille, A., Ravichandran, A., Bhotika, R., Soatto, S.: Predicting training time without training. Advances in Neural Information Processing Systems **33**, 6136–6146 (2020)
57. Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A.M., Le, Q.V., Laudon, J., et al.: Mixture-of-experts with expert choice routing. Advances in Neural Information Processing Systems **35**, 7103–7114 (2022)

A Diffusion Soup: Model Merging for Text-to-Image Diffusion Models (Supplementary Material)

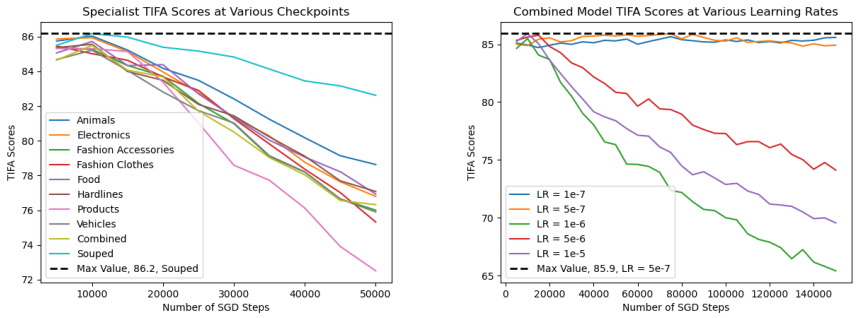

Fig. 7: (Left) Diffusion Soup Outperforms All Models at All Checkpoints. We show that optimizing over the number of SGD steps at LR = 1e-6 demonstrates that at all checkpoints, the souping model outperforms all individual models, as well as the combined models. The shallow dropoff in performance of the souped model relative to the steep dropoff of individual models at larger steps is indicative of strong robustness properties of Diffusion Soup. **(Right) Souping Outperforms the Paragon even when Optimizing the Latter for LR and Step Size Jointly.** We show that optimizing over Learning Rates and Step size jointly yields a better Combined Model Paragon (LR = 5e-7, Num Steps = 80k) than the original used in the main text (LR = 1e-6, Num Steps = 10k). However, Diffusion Soup still outperforms (TIFA 85.9 $\rightarrow$ 86.2) this new Combined Model Paragon checkpoint.

The supplementary material is organized as follows. Appendix B provides further details about hyperparameters used to conduct experiments in the paper, alongside ablations that justify our choice of hyperparameters. Appendix C provides a comparison of Diffusion Soup to a computationally expensive Ensemble model. Appendix F provides additional examples of style mixing using compositions of famous artistic styles. Appendix G provides additional examples of greedy souping. Finally, Appendix J gives proofs for the main propositions of the paper.

B Choices of Hyperparameters and Ablations

The primary focus of our work is to showcase the benefits of Diffusion Soup atop a *pre-existing* collection of finetuned checkpoints. Specifically, we aim to demonstrate that regardless of the ingredients provided, souping improves performance over any individual model, or the traditional approach of aggregating datasets to build one model (the Combined Paragon model). In this section, we show that regardless of the finetuned checkpoints, souping outperforms its

constituent ingredients. We further optimize learning rates to find the strongest possible combined model paragon, to show the dominance of the souping approach even when the paragon is hyperparameter tuned at a disadvantage to souping.

To train our finetuned checkpoints for both the Specialist experiments and the Aesthetic Enhancement experiments, we use the AdamW optimizer with a Learning Rate of 1e-6. Each of these finetuning runs takes 3 hours to complete on a server with 8x 40GB A100 GPUs. Our choice is mainly motivated by the Emu Paper [11] who recommend a low learning rate and finetuning for up to 15k steps to prevent model memorization and forgetting. We ablate the choice of the number of steps in Fig. 7 (Left), and come to several conclusions. First, we show that as [11] suggest, the dropoff point for all finetuned models begins around 15k steps, and all models, including the Combined model peak at 10k steps. We thus use the 10k checkpoints for all of our individual and souping experiments in the main text. Moreover, we show that regardless of the number of steps chosen, the souped model always outperforms the constituent ingredients *and* the combined model paragon, a powerful result that is indicative of the robustness of the souping approach. The horizontal black bar demonstrates that souping achieves the highest value overall, a TIFA score of 86.2, the number we report in the main text for Uniform Souping.

We further ablate the choice of learning rate and step size for the Combined model Paragon in Fig. 7 (Right), to ensure that souping is being compared to the strongest possible paragon baseline. We find that a combination of an even lower learning rate (5e-7), and an extremely large step size of 80k steps outperforms ($85.5 \rightarrow 85.9$) the combined model paragon at the learning rate of 1e-6 and step size of 10k that we use in the main text. However, our souped model at 1e-6 and step size of 10k still outperforms ($85.9 \rightarrow 86.2$) this new Combined model checkpoint. We thus show that our souping methodology and findings are robust to hyperparameter tuning.

C Comparison to Ensembling

In this section we compare Diffusion Soup against the *ensembling* baselines [14, 18, 30]. These methods train different models on different subsets of data, and merge the flows—outputs of the denoising network, rather than the weights of the network as in souping—at inference. Precisely, ensembling approaches compute the output of the denoising network of each ingredient model at each time-step during backward diffusion and average them using weighted coefficients. We follow the approach in [18] which shows that the weighted coefficients can be computed using a trained classifier.

Critically, ensembling approaches require *each ingredient model to be loaded and run at inference time*. Therefore, they have a much larger memory footprint, and much greater inference overhead than Diffusion Soup, which generates a single model for use at inference time. To provide an anecdote illustrating the difference, in this paper we run all of our experiments on a single p4d.24xlarge

Table 3: Diffusion Soup is Competitive even with a Computationally Expensive Ensembling Approach. We revisit the setting of Table 2 in the main text, taking the first two aesthetic subsets AE1 and AE2, in order to compare Diffusion Soup with an ensembling approach. As observed previously, souping outperforms each constituent model and the combined paragon. We further show that the performance of our model is even competitive with a computationally costly Ensemble paragon, which computes the outputs of the denoising network using every model at every time step. The relative closeness of the two approaches, and the computational feasibilty of Diffusion Soup makes it a particularly attractive choice for practical users of compartmentalization.

		TIFA	IR	CLIP
Base	SD2.1	85.5	0.30	0.259
SFT	AE1	86.5	0.53	**0.263**
	AE2	86.3	0.54	0.261
Combined (Paragon)	AE1 & AE2	86.0	0.40	0.261
Ensemble (Paragon)	AE1 & AE2	**87.3**	**0.64**	**0.263**
Souped (Ours)	AE1 & AE2	86.7	0.60	**0.263**

EC2 instance in the AWS Cloud. Whereas our souping experiments often aggregate 5-9 model checkpoints together, we can only ensemble with 2 models without running into OOM errors on an A100 GPU.

We work within these limitations in Tab. 3, where we compare our method against the ensembling baselines. We take a pre-trained Stable-Diffusion model and fine-tune it in on two subsets of the MS-COCO dataset. Diffusion Soup is within 1% (86.7 $\rightarrow$ 87.3) TIFA and 93.75% IR (0.6 $\rightarrow$ 0.64) to ensembling. The small performance gap between the two approaches, and the computational feasibilty and scalability of Diffusion Soup makes souping a particularly attractive choice for practical users of compartmentalization.

D Diffusion Soup is Agnostic to the Base Model Checkpoint

In the main text, our experiments exclusively leverage a pretrained Stable Diffusion 2.1 (SD 2.1) as the base model checkpoint, upon which SFT is performed on various shards, and the resulting model checkpoints are souped together. Our findings are not unique to SD2.1, and broadly apply to any Diffusion Model, a point demonstrated concisely in the following table.

Table 4 extends the setting of the main paper's Table 2 to two new architectures, Stable Diffusion XL (SDXL) [38] and PixArt-α (PixArt) [6]. Here, we perform SFT on two random aesthetic shards, Shard 1 and Shard 2 and display TIFA scores for the base models, each shard, the combined model paragon, and the model soup. For consistency with the main paper, we finetune and evaluate TIFA at SD 2.1's native 512 x 512 resolution. For both architectures, souping once again bests all base models, models finetuned on individual shards, and the

Table 4: Diffusion Soup is Agnostic to the Base Model Checkpoint. We revisit the setting of Table 2 in the main text and use two random aesthetic shards to perform SFT, Shard 1 and Shard 2. We compute TIFA scores to measure text-to-image alignment in order to evaluate the performance of Diffusion Soup when the base model checkpoint is SDXL and PixArt. As observed previously, souping outperforms the base model, each constituent model and the combined paragon. Of particular significance is PixArt, since it is a diffusion transformer (DiT) rather than a U-Net based diffusion model. These findings suggest that the strong performance of Diffusion Soup holds across different architectures.

		TIFA
SDXL	Base	81.2
	Shard 1	85.2
	Shard 2	85.5
	Combined (Paragon)	85.5
	Souped (Ours)	**85.7**
PixArt	Base	87.2
	Shard 1	86.9
	Shard 2	87.4
	Combined (Paragon)	87.4
	Souped (Ours)	**88.1**

combined model trained on a union of the shards. Our results demonstrate the same strong performance in text-to-image alignment found in the main text for SD 2.1, and suggest that souping indeed generalizes across architectures. Our PixArt results stand out in particular, since it is a diffusion transformer (DiT) rather than a U-Net.

E Souping Requires Models Finetuned from a Shared Pretrained Checkpoint

Throughout the paper, we leverage a common pretrained checkpoint, SD 2.1, on which SFT is performed on various shards prior to souping. In this section, we demonstrate that the constituent models used for souping must stem from a shared pretrained checkpoint. That is, souping two models trained on different datasets from an identical *random* initialization does not work. In figure 8, we consider the output of a publicly released SD 2.1 checkpoint (left), and an SD 2.1 model trained on an internal dataset with identical initialization (middle). When provided the same prompt, the models produce similar images. However, when souped (right), the models produce noise as output. This finding suggests that souping requires models finetuned from a shared pretrained initialization, and leverages the approximate linearity of the SFT checkpoints in the local neighborhood of the pretrained initialization.

Fig. 8: Souping Requires Models Finetuned from a Shared Pretrained Checkpoint. We show that when souping two models that have been trained from an identical random initialization, the result is random noise. (Left) The output of the publicly released SD 2.1 model for the prompt "a photo of an office with a couch". (Middle) The output of an SD 2.1 model trained on an internal dataset with the same random initialization, and on the same prompt. (Right) The output of the souping the two models on the same prompt, which resembles random noise. This finding is indicative of model souping's strong dependence on the linearity of finetuned checkpoints within the local neighborhood of a shared pretrained initialization.

F Additional Examples of Style Mixing

We present additional zero-shot style mixing results using models fine-tuned on *Ukiyo-e* and *Romanticism* images from the WikiArt dataset [48]. Fig. 9 demonstrates souping Ukiyo-e and Romanticism styles. Fig. 10 demonstrates blending the Ukiyo-e style with Pokemon [36] and FSCOCO [10] styles. The final row of this figure demonstrates souping *all three* styles.

G Greedy souping for *Enhancing Image Aesthetics*

We extend the experimental section from Sec. 6.2 and Tab. 2 with Greedy Souping results (see Sec. 4.2). The original cumulative findings showed that Uniform Soup outperforms its ingredients in a cumulative sense, i.e. as the number of checkpoints grow from AE1 all the way to AE1-AE5. Our greedy approaches strategically select a subset of these checkpoints to show that performance can be further optimized in TIFA score. Identical to the setting with Specialists, Greedy Souping outperforms all other souping approaches, and both Greedy Souping and Reverse Greedy Souping are the two highest performing approaches. Our strong performance demonstrates that Greedy Souping offers benefits when merging models trained on a common task (in this case aesthetic enhancements) as well as aggregating specialists (see Tab. 1).

Table 5: Greedy Souping Methods Outperform all Approaches in Aesthetic Enhancement. We once again consider the setting of the main paper's Table 2, with data shards that all have one task in common, aesthetic enhancement. This time, we add our Greedy Souping approaches to Table 2 (R. refers to Reverse), showing that they outperform all other methods in Aesthetic Enhancement. These findings replicate the strong performance of Greedy Approaches found in aggregating specialists in the main text's Table 1.

	Base & SFT			Combined (Cumulative)			Souped (Cumulative)		
	TIFA	IR	CLIP	TIFA	IR	CLIP	TIFA	IR	CLIP
SD2.1	85.5	0.30	0.259	-	-	-	-	-	-
AE1	86.5	0.53	**0.263**	86.5	0.53	**0.263**	86.5	0.53	**0.263**
AE2	86.3	0.54	0.261	86.0	0.40	0.261	86.7	**0.60**	**0.263**
AE3	86.1	0.47	0.262	85.7	0.38	0.261	86.8	**0.60**	**0.263**
AE4	86.2	0.43	0.260	86.0	0.39	0.261	86.8	0.59	**0.263**
AE5	85.9	0.45	0.262	85.6	0.37	0.261	86.8	0.59	**0.263**
Greedy Soup	-	-	-	-	-	-	**87.0**	**0.60**	**0.263**
R. Greedy Soup	-	-	-	-	-	-	86.9	0.59	**0.263**

H Quantitative Analysis of Style Mixing

While the main text showcases qualitative results of Style Mixing, this section aims to quantify the ability of Diffusion Soup to capture different stylistic elements *relative* to existing methods. In order to baseline the performance of Diffusion Soup, we consider two variants of Textual Inversion [15], a method which tunes a new style token S on a reference dataset in order to capture its style and extend it to new prompts. The first variant, *TextInv-Union* finetunes S on a concatenation of datasets of different styles so that S captures a mixture of two styles. The second, *TextInv-Avg* obtains S by finetuning on each style dataset and averaging them together. To quantify the performance of style mixing, we consider average CLIP score between generated images and a textual style description, which is found by showing a Vision Language Model 5 random images from the training shards and asking for a stylistic description. Our results, displayed in Table 6 demonstrate the supremacy of model souping over both of these baselines, indicating that the qualitative results found in the main text are indeed the product of strong style mixing capabilities.

I Souping can Approximate the Arithmetic Mean

In the main text, we show that souping with fixed weights, $k_i = 1/n$ approximates the Geometric Mean. In this section, the following proposition we show that with a carefully chosen k_i, souping can theoretically sample from the union of all data. As before, the proof of the proposition is in the proofs section of the appendix.

Proposition 3. *(Arithmetic Mean) Let* $\nabla_{x_t} \log p^{(i)}(x_t)$ *be the marginal score in Eq.* (2) *where* $x_t = \gamma_t x_0 + \sigma_t \epsilon$*, with* $x_0 \sim p^{(i)}(x_0)$ *and* $\epsilon \sim \mathcal{N}(0, I)$*. Let* $\epsilon_{w_i}(x_t, t, y)$

Table 6: Diffusion Soup Quantatively Outperforms Baselines in Style Mixing. We compare two variants of Textual Inversion [15] adapted for style mixing to Diffusion Soup. We leverage Average CLIP score between generated images and a VLM generated textual style description in order to quantify how well the generated images capture each style. Our baselines consist of (1) *TextInv-Union*, which finetunes the Textual Inversion token S on a combination of the style datasets, and (2) *TextInv-Avg*, which averages the Textual Inversion token embeddings finetuned independently on each dataset. The Average CLIP Scores shown below highlight the strong quantative performance of Style Mixing relative to two style mixing baselines.

Style	Soup (**Ours**)	TextInv-Union	TextInv-Avg
Pokemon	**0.169**	0.151	0.163
FSCOCO	**0.194**	0.142	0.184

be a neural network (with sufficient capacity) trained to match $\nabla_{x_t} \log p^{(i)}(x_t)$. *Sampling using* $(1/n)\sum_{i=1}^{n} \lambda_i \frac{p^{(i)}(x_t)}{p(x_t)} \nabla_{x_t} \log p^{(i)}(x_t)$ *with Eq.* (2) *generates samples from* $\sum_i \lambda_i p^{(i)}(x_0)$, *where* $p(x_t) = \sum_i \lambda_i p^{(i)}(x_t)$, $\sum_i \lambda_i = 1$. *Furthermore, using Eq.* (4), ϵ_{soup} *with* $k_i = \lambda_i \frac{p^{(i)}(x_t)}{p(x_t)}$ *generates samples from* $\sum_i \lambda_i p^{(i)}(x_0)$.

In combination with Proposition 1, depending on the choice of k_i in souping, we show that our model can sample from different distributions. Note that in Proposition 3, k_i depends on the t and thus requires re-souping per prompt at every timestep. In this work, k_i is fixed across prompts and timesteps, and set to be uniform ($k_i = 1/n$) or chosen using various greedy approaches.

J Proofs

We restate each proposition for convenience prior to providing their proof.

J.1 Proposition 1

Proposition 1. *(Geometric Mean) Let* $\nabla_{x_t} \log p^{(i)}(x_t)$ *be the marginal score (in Eq.* (2)*) where* $x_t = \gamma_t x_0 + \sigma_t \epsilon$, *with* $x_0 \sim p^{(i)}(x_0)$ *and* $\epsilon \sim \mathcal{N}(0, I)$. *Let* $\epsilon_{w_i}(x_t, t, y)$ *be a neural network (with sufficient capacity) trained to match* $\nabla_{x_t} \log p^{(i)}(x_t)$. *Under certain conditions on the sampling procedure Eq.* (2), *using* $(1/n)\sum_{i=1}^{n} \nabla_{x_t} \log p^{(i)}(x_t)$ *generates samples from* $Z^{-1}\big(\Pi_i p^{(i)}(x_0)\big)^{1/n}$. *Furthermore, using Eq.* (4), ϵ_{soup} *with* $k_i = 1/n$ *generates samples from* $Z^{-1}\big(\Pi_i p^{(i)}(x_0)\big)^{1/n}$.

Proof Let $p_{\text{geometric}} = Z^{-1}\big(\Pi_i p^{(i)}(x_0)\big)^{1/n}$ be the geometric mean of the initial set of distributions provided to us. The marginal distribution at time t corresponding to the geometric mean is given by $p(x_t) = \int Z^{-1}\big(\Pi_i p^{(i)}(x_0)\big)^{1/n} p(x_t|x_0) dx_0$. Sampling from this distribution requires access to the score of the marginal,

which is usually difficult to obtain. This score is given by:

$$\nabla_{x_t} \log p(x_t) = \nabla_{x_t} \log \int Z^{-1} \left(\Pi_i p^{(i)}(x_0)\right)^{1/n} p(x_t|x_0) dx_0$$

where the integral inside the logarithm is difficult to estimate. However, we can show that we can construct a simple score function which is easy to estimate and can generate samples from $p_{\text{geometric}}$:

$$\nabla_{x_t} \log p'(x_t) = (1/n) \sum_i \nabla_{x_t} \log \int p(x_t|x_0) p^{(i)}(x_0) dx_0$$

$\nabla_{x_t} \log p'(x_t)$ can be used for sampling because of the fact that it generates a sequence of distributions where its equal to $\nabla_{x_t} \log p(x_t)$ at $t = 0$, and $\mathcal{N}(0, I)$ at $t = T$. Thus using $\nabla_{x_t} \log p'(x_t)$ we can create a sequence of distributions which generates a sample from $p_{\text{geometric}}$ as $t \to 0$ during backward diffusion using Eq. (2). Note that this sequence of distributions is different from the sequence of distributions generated by the forward diffusion process. Convergence of langevin based MCMC (using Eq. (2) for sampling) is a very well studied problem [8,35,49, 55], showing that using Eq. (2) indeed converges to stationary distribution given by $p_{\text{geometric}}$ in this case. Furthermore, [55] in Theorem 1 shows that sampling using Langevin dynamics (Eq. (2)) with a sufficiently powerful score estimator (in our case a diffusion model) generates samples from the initial distribution, $p_{\text{geometric}}$. This proves the first part of the result. The second part of the proof is a mere application of Eq. (4) and Eq. (5) i.e. we replace each $\nabla_{x_t} \log p^{(i)}(x_t)$ with $-\epsilon_{w_i}(x_t, t, y)$. More precisely,

$$\nabla_{x_t} \log p(x_t) = (1/n) \sum_i -\epsilon_{w_i}(x_t, t, y) = -\epsilon_{\text{soup}} \tag{8}$$

J.2 Proposition 2

Proposition 2. *(ε-NAF for Diff. Soup) Let $\epsilon_{soup}(x_t, t, y) \triangleq \epsilon_{\frac{1}{2}(w_1+w_2)}(x_t, t, y)$ be the soup model obtained by training w_i on D_i, such that $D = D_1 \cup D_2$, and $D_1 \cap D_2 = \varnothing$. Then, under certain conditions sampling, using $\epsilon_{soup}(x_t, t, y)$ satisfies ε-NAF for some ε which depends only on D_1 and D_2.*

Proof [50] shows that given two generative models $p_1(x|y), p_2(x|y)$, sampling from $\sqrt{p_1(x|y)p_2(x|y)}/Z$ produces data which is NAF with respect to some protected samples in the training data. We need to show that souping generates samples from $\sqrt{p_1(x|y)p_2(x|y)}/Z$. Note that this directly follows from Proposition 1 for the case $n = 2$.

J.3 Proposition 3

Proposition 3. *(Arithmetic Mean) Let $\nabla_{x_t} \log p^{(i)}(x_t)$ be the marginal score (in Eq. (2)) where $x_t = \gamma_t x_0 + \sigma_t \epsilon$, with $x_0 \sim p^{(i)}(x_0)$ and $\epsilon \sim \mathcal{N}(0, I)$.*

Let $\epsilon_{w_i}(x_t, t, y)$ be a neural network (with sufficient capacity) trained to match $\nabla_{x_t} \log p^{(i)}(x_t)$. Sampling using $\sum_{i=1}^{n} \lambda_i \frac{p^{(i)}(x_t)}{p(x_t)} \nabla_{x_t} \log p^{(i)}(x_t)$ with Eq. (2) generates samples from $\sum_i \lambda_i p^{(i)}(x_0)$, where $p(x_t) = \sum_i \lambda_i p^{(i)}(x_t)$, $\sum_i \lambda_i = 1$. Furthermore, using Eq. (4), ϵ_{soup} with $k_i = \lambda_i \frac{p^{(i)}(x_t)}{p(x_t)}$ generates samples from $\sum_i \lambda_i p^{(i)}(x_0)$.

Proof Let $\nabla_{x_t} \log p(x_t)$ be the score the mixture of distributions $p(x_t) = \sum_i \lambda_i p^{(i)}(x_t)$. We show that the score of the mixture is a convex combination of the individual scores:

$$\begin{aligned}
\nabla_{x_t} \log p(x_t) &= \nabla_{x_t} \log \sum_i \lambda_i p^{(i)}(x_t) \\
&= \frac{1}{\sum_i \lambda_i p^{(i)}(x_t)} \sum_i \lambda_i \nabla_{x_t} p^{(i)}(x_t) \\
&= \frac{1}{\sum_i \lambda_i p^{(i)}(x_t)} \sum_i \lambda_i \frac{p^{(i)}(x_t)}{p^{(i)}(x_t)} \nabla_{x_t} p^{(i)}(x_t) \\
&= \frac{1}{\sum_i \lambda_i p^{(i)}(x_t)} \sum_i \lambda_i p^{(i)}(x_t) \nabla_{x_t} \log p^{(i)}(x_t) \\
&= \sum_i \frac{\lambda_i p^{(i)}(x_t)}{\sum_i \lambda_i p^{(i)}(x_t)} \nabla_{x_t} \log p^{(i)}(x_t) \\
\implies \nabla_{x_t} \log p(x_t) &= \sum_i \frac{\lambda_i p^{(i)}(x_t)}{\sum_i \lambda_i p^{(i)}(x_t)} \nabla_{x_t} \log p^{(i)}(x_t)
\end{aligned}$$

Hence, we show the first part of the proof. The second part of the proof is a mere application of Eq. (4) and Eq. (5) i.e. we replace each $\nabla_{x_t} \log p^{(i)}(x_t)$ with $-\epsilon_{w_i}(x_t, t, y)$:

$$\begin{aligned}
\nabla_{x_t} \log p(x_t) &= \sum_i \frac{\lambda_i p^{(i)}(x_t)}{\sum_i \lambda_i p^{(i)}(x_t)} (-\epsilon_{w_i}(x_t, t, y)) \\
&= -\epsilon_{\text{soup}}
\end{aligned} \tag{9}$$

where we use Eq. (5) to show the previous equality with $k_i = \dfrac{\lambda_i p^{(i)}(x_t)}{\sum_i \lambda_i p^{(i)}(x_t)}$.

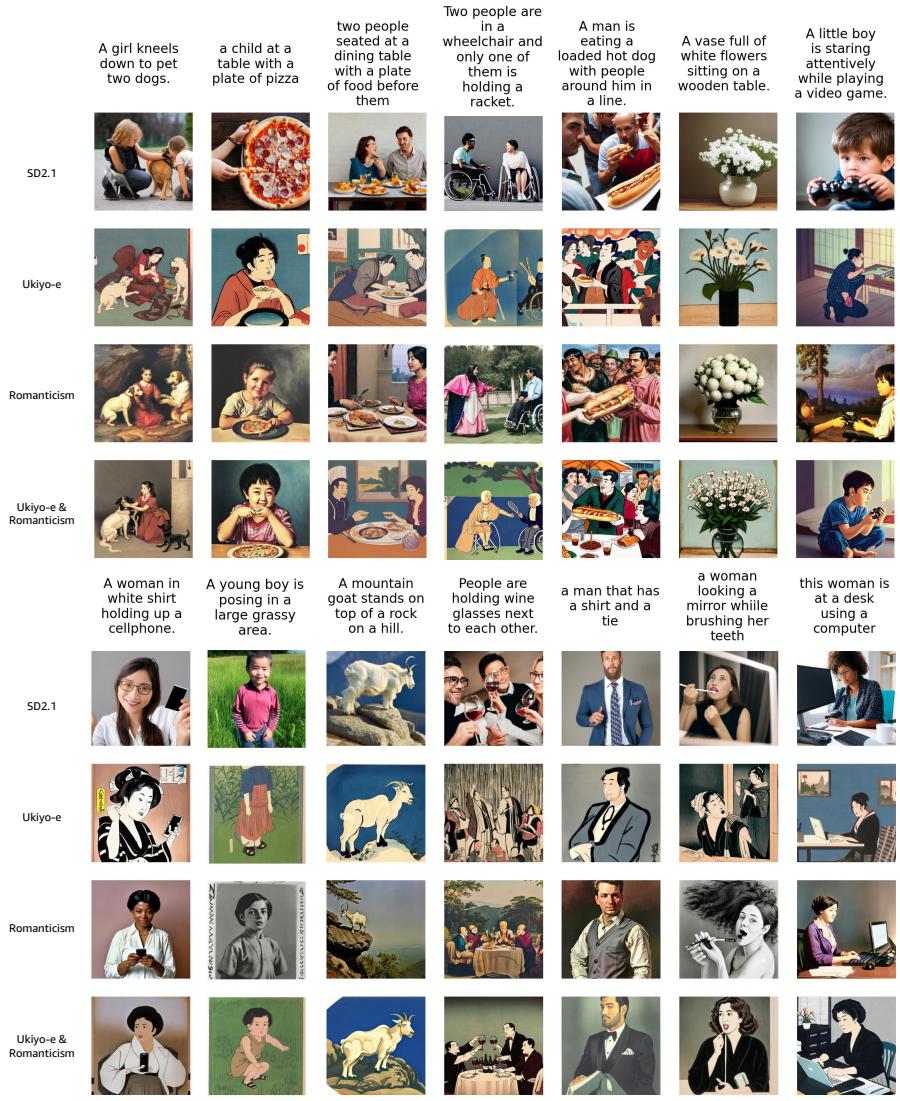

Fig. 9: Diffusion Soup merges finetuned models to create hybrid artistic styles. We apply Diffusion Soup to the SD2.1 model (*Row 1*) finetuned on Ukiyo-e (*Row 2*) and Romanticism (*Row 3*) styles from WikiArt [48] to create a hybrid style (*Row 4*). The results are zero-shot since we do not have examples of the hybrid style for training.

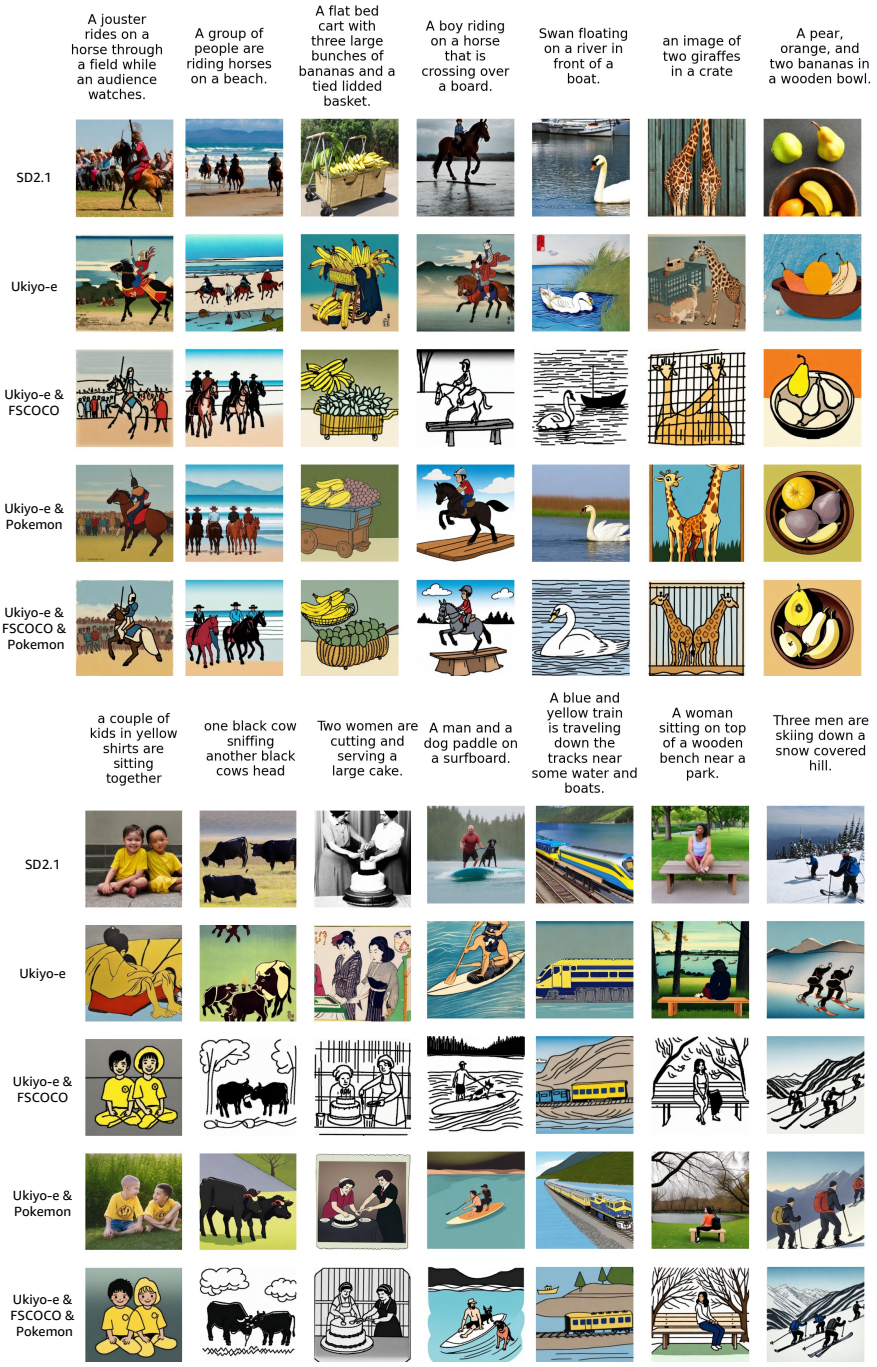

Fig. 10: Diffusion Soup merges *three* finetuned models to create hybrid styles. We apply Diffusion Soup to the SD2.1 model (*Row 1*) finetuned on Ukiyo-e (*Row 2*) to create various hybrid styles. Ukiyo-e is souped with FSCOCO (*Row 3*), Pokemon (*Row 4*) and both models (*Row 5*).