A. Overview

In Appendix B, we consider a baseline for domain expansion and demonstrate it is inferior to our proposed method. Next follows the main part of the supplementary, Appendix C, in which we perform additional analysis and experimentation of our method. Finally, in Appendix D, we provide additional details completing the paper.

B. Domain Expansion Baseline Using Class-Conditioning

In this section, we experiment with an alternative, baseline, method to perform domain expansion. Generative models capturing multiple domains commonly use a class-conditioning mechanism [3]. Adopting this approach, we attempt to perform domain expansion by modeling domains with classes. We find that this method does not work as well as our proposed method.

Method. We start with an unconditional pretrained generator, specifically StyleGAN [13]. We then make the generator condition on a one-hot vector, using the architecture proposed by Karras et al. [11]. This change involves adding a single MLP layer, whose input is the one-hot vector. Its output is concatenated to the random latent code and then fed to the generator.

The class-conditioned generator is trained in a similar protocol to our method. The source domain uses class \(c = 0\), which is analogous to the base subspace. Whenever the 0th class is sampled, we apply the original loss \(L_{\text{src}}\) and the memory replay regularization (See Sec. 3.3). Formally, the loss describing this training is

\[
L_{\text{reg}} = \mathbb{E}_{z \sim p_{\text{src}}(z)}\left[\lambda_{\text{src}} L_{\text{src}}(G(z, c = 0)) + L_{\text{recon}}(G(z, c = 0))\right],
\]

where \(L_{\text{recon}}\) is the memory-replay loss defined in Eq. (5) and \(\lambda_{\text{src}} = 1\) is a hyperparameter weighting the losses. Other classes, analogous to repurposed subspaces, are dedicated to the newly introduced domains. Whenever the \(i\)th class is sampled \((i > 0)\), we apply the loss of the domain adaptation task \(L_i\). Applied over all new domains, the expansion loss is formally given by

\[
L_{\text{expand}} = \sum_{i=1}^{N} \mathbb{E}_{z \sim p_i(z)} L_i(G(z, c = i)).
\]

The final training objective still reads as \(L_{\text{full}} = L_{\text{expand}} + L_{\text{reg}}\).

Experiments. We expand an FFHQ [12] generator with two new domains, “Sketch” and “Tolkien Elf”, introduced using StyleGAN-NADA [5]. We display the generated images using the same \(z\) latent codes for the different classes Fig. 1a.

We qualitatively observe that the expanded, class-conditioned generator preserves the source domain well, also expressed by preserving the FID [7] score. However, for new domains, we observe degraded performance from two aspects. First, the class-conditioned generator “leaks” knowledge between the classes. For example, in Fig. 1a, faces generated from the class dedicated to sketches also have long, elf-like, ears. Second, the domains are not “aligned”. Despite being generated from the same \(z\) latent codes, the images differ beyond the differences between domains. For example, corresponding images from the source domain and elf domain often portray different head poses and facial expression. Therefore, it is not clear how can one obtain the elf “version” of a given face image, limiting the applications of such a model.

For reference, we display comparable results from our expansion method in Fig. 1b. As can be seen, our method does not suffer from these issues.

C. Additional Experiments

C.1. Latent Directions Analysis

Our method explicitly relies on the existence of dormant directions and their distinction from non-dormant directions. We wish to emphasize that the dichotomous distinction between “dormant” and “non-dormant” is a simplification. In Fig. 2, we report the mean LPIPS distance induced
Figure 1. Experimenting with a class-conditioned baseline for domain expansion. (a) Images generated from a class-conditioned expanded model from the same z latent codes for the source, sketch, and elf domains. The source domain is preserved well in its dedicated class. However, the newly introduced domains “leak” information, expressed in long, elf-like, ears in the sketch domain. Additionally, the different domains are not well-aligned, as changing the domain also results in unrelated changes to head pose and facial expressions. (b) Comparable results from our domain expansion method, provided for reference. As can be seen, using our method, the domains do not interfere with each other and are well-aligned.

to images by a $3\sigma$ traversal along each direction. As can be seen, the distance is never exactly 0 and there is also no clear discontinuity. Nevertheless, it is clear that later directions, usually those beyond 100, cause significantly smaller perceptual change in the generated image. This behavior can also be qualitatively observed in Fig. 3.

As discussed in Sec. 4.1, this “almost” monotonous behavior is expected as our latent directions are right-singular vectors, sorted in decreasing order according to their corresponding singular values [36].

C.2. Effect of Choice of Direction for Domain

Our method dedicates a single dormant direction for every newly introduced domain. As mentioned in Sec. 4.1, all previous experiments used the last dormant directions, sorted in decreasing order according to their corresponding singular values. One might wonder: Why should one use the last directions? And among the last directions, how should one match a direction to a domain?

We now demonstrate that the specific choice of a latent direction has no significant impact on results, as long as it is dormant. To this end, we perform multiple expansions, each with 5 new domains introduced by StyleGAN-NADA [5], starting from a single generator pretrained on AFHQ [4]. For 4 of the new domains – “Siberian Husky”, “Pixar”, “Funny Dog”, “Boar” – we dedicate the same directions in all experiments. Specifically, we use directions 507 – 510, respectively. Directions numbers refer to their location in the decreasingly sorted right-singular vector set. Recall that the dimension of the latent space is 512, hence these directions are among the last ones. For the last domain, “Sketch”, we vary the dedicated direction, using one of the directions 200, 300, 400, 500, 511. We run the expansion twice with different random seeds.

We study how the choice of direction for the Sketch domain affects its performance. In Fig. 4 (top) we report the CLIP error of images generated from the “Sketch” subspace with the prompt “a sketch” as a function of training iterations. We additionally display sample of generated images from each model in Fig. 4 (bottom). As can be seen, similar results are produced from different repurposed directions. Specifically, visual differences observed using different directions, are similar to those observed using the same directions but with different random seeds. This indicates that the differences between directions are negligible and might be entirely due to random chance.

Nevertheless, we do observe that certain directions minimize the CLIP error slightly more efficiently, across random seeds. We therefore run additional 5 expansions, using “Bear” instead of “Sketch”. We now observe a different ordering of directions. We therefore conclude, that even if slight, imperceptible, differences exist between directions, they are not consistent across domains.

In summary, the choice of dormant direction has little to no effect. This result is arguably intuitive, as all dormant directions might be considered equivalent, having insignificant effect on generated images. Therefore, our choice of using the last directions is almost arbitrary, only motivated by the fact that they are the “most dormant”. Similarly, no technique is required to match an direction to a domain, and one can simply pick a dormant direction randomly.

C.3. Repurposing Non-Dormant Directions

Aiming at domain expansion, preserving the source domain is integral. Since the non-dormant directions span the variations of the source domains, we explicitly kept them
The capability to add new concepts in addition to existing domain expansion. Previous results are clearly not solving domain expansion, as they alter the original behavior of the source domain. Instead, one might say they adapt the domain modeled by the generator. Nevertheless, there exists a profound difference to existing domain adaptation methods. Our resulting generator does not completely overriding the source domain. Instead, in a precise and controllable manner, it modifies individual factors of variation. Therefore, a user can carefully \textit{rewrite} \citep{1,43} the semantic rules of a generative model, allowing greater control and freedom.

\subsection*{C.4. Distance to Repurposed Subspace}

Repurposed subspaces are defined by transporting the base subspace along a linear direction by a predetermined scalar size \(s\) (See Eq. (3) in the main paper). All results in the paper, across domains and variations used \(s = 20\). We next evaluate the effect the hyperparameter \(s\) has on results. To this end, we perform multiple expansions of an FFHQ \citep{12} generator with 100 new variations, while varying the value of \(s\).

We measure CLIP errors (introduced in Sec. 4.3) of images generated from repurposed subspaces and the corresponding target text used for training, as a function of training iterations. In Fig. 7a we report the results for two variations - “Marge Simpson” and “Tolkein Elf”. As can be seen, for all \(s > 0\), CLIP error decreases as training progresses, and it decreases “faster” for greater values of the parameter \(s\). Even with \(\times 10\) more iterations, the model trained with \(s = 5\) does not reach the CLIP error of the model trained with \(s = 20\).

Images generated from the repurposed subspaces are displayed in Fig. 7b. For each value of \(s\), we use the checkpoint that resulted in the closest CLIP error to that obtained by a favored \(s = 20\) checkpoint. As can be seen, not only training time is affected by parameter \(s\), but the visual effects captured by training vary significantly.

We observe that models trained with greater values of parameter \(s\) depict a more significant change with respect to the source domain. When parameter \(s\) is too small (e.g., \(s \leq 5\), the model captures only few, simple characteristics of the new domain. On the other hand, when parameter \(s\) is too large (e.g., \(s = 50\)), the model commonly generates images that are blurry, have color artifacts or even do not capture the target text well. For example, with the target text “Marge Simpson”, the model learns to generate images with blue skin rather than blue hair. We note that these undesired artifacts cannot be mitigated by training with a large value of parameter \(s\) originally, and use a smaller one in test-time,
Figure 3. Visualization of $\pm 3\sigma$ traversal along latent directions in the FFHQ [12] and LSUN Church [48] models, obtained using SeFA [36]. Directions shown are sorted from least ($v_0$, top) to most ($v_{511}$, bottom) dormant. As can be seen, later directions are dormant – not affecting the generated image. We over-sample early directions for clarity. In practice, over 80% of directions are dormant.
Figure 4. We expand a generator pretrained on AFHQ [4] with 5 domains, varying the dormant direction dedicated to the “sketch” domain. We repeat the expansion twice, with different random seeds. Top - reporting CLIP error of images generated from the sketch domain with the text “a sketch”. Bottom - a sample of generated images from checkpoints obtaining CLIP error closest to the horizontal black line. As can be seen, images generated using different repurposed dimensions differ only slightly. Specifically, changing the random seed induces similar difference.

Figure 5. Similar to Fig. 4, using a “bear” domain instead of “sketch”. As can be seen, dimensions are ordered differently in terms of minimizing CLIP error, as compared to their order for sketch.

as demonstrated in Fig. 8.

Following these results, we conclude that the parameter $s$ has a regularizing effect. Placing the domains “closer” in the latent space causes them to be more similar in image space as well. Conversely, placing the domains further apart allows the new domain to capture more drastic, out-of-domain effects.

Eventually, choosing a value for parameter $s$ is subject to user preference. In our experiments, we have found that values in the range of $\{10, 30\}$ offer satisfying results, across different source and expanded domains.

We last note that the regularization effect of parameter $s$ could be explained by the existence of a globally consistent “pace of change” of the generator with respect to the latent space. With StyleGAN, such behavior is explicitly encouraged using a Perceptual Path Length (PPL) regularization term [13]. Nevertheless, we observe identical results when omitting this regularization during our expansion.

C.5. How Many Domains Can Fit?

So far, the largest number of new domains used for expansion was 105. The results from Appendix C.1 indicated that there might be up to 400 dormant directions. Could they all be repurposed?

We apply our method to expand a generator pretrained on FFHQ with 400 new domains, repurposing the last (and perhaps all) dormant directions. Incredibly, the expansion succeeds. We find that the expansion follows the same findings discussed in Sec. 4.3 – training is slower, yet quality is uncompromised. Specifically, the FID score from the base subspace is 2.83 compared to 2.80 in our model expanded with 105 domains. We display images generated from this model in the accompanying video and in Figs. 9 to 11.
D. Additional Details

D.1. Training Time and Iterations

When expanding the generator with a single new domain, our training requires roughly twice the number of iterations to obtain comparable effects. The difference is a direct result of our additional regularization terms. With additional domains, we observe a roughly linear relationship between the number of domains and the required training iterations. For example, the FFHQ model expanded with 105 iterations was trained for $40K$ iterations, while the model with 400 iterations was trained for $150K$ iterations.

Note that different training objective might require a different number of iterations. StyleGAN-NADA [5] specifically heavily relies on early-stopping. An ideal domain expansion method could consider this issue, and sample training objectives to apply non-uniformly. In practice, we did not observe this to be an issue, probably due to our method optimizing numerous objectives simultaneously.

D.2. Transformation of Loss Function

As explained in Sec. 3.2, transforming a given domain adaptation task to perform domain expansion requires limiting the samples latent codes. The loss function itself, in principal, is left unchanged. This is exactly the case for MyStyle [21]. For StyleGAN-NADA [5], however, we made a subtle modification to the loss function.

StyleGAN-NADA computes its loss with respect to a frozen copy of the source generator (See Sec. 4.1). This is done in order to maintain access to the source domain, despite it vanishing from the adapted generator during training. Conversely, using our method, the source domain is preserved along the base subspace. We take advantage of this fact and modify the loss only slightly. Instead of using a frozen generator to generate images from the source domain, we simply use our expanded generator and latent codes from the base subspace.

C.6. Additional Compositions Results

In Figs. 12 and 13 we provide additional qualitative results displaying compositionality in expanded generators.

Figure 6. Using our training method with non-dormant direction rewrites existing semantic rules and adds new concepts on top of existing ones. (a) Traversing the $1^{st}$ direction originally made people older and more masculine. After fine-tuning, it also adds a beard. (b) Traversing the $8^{th}$ direction originally turned people heads. After fine-tuning it also turns them to elves.
Figure 7. Evaluating the effect of the distance between the base and repurposed subspace, $s$. (a) We compare CLIP error as a function of training iterations, between models trained with different values of parameter $s$. (b) Generated images from models having CLIP error as close as possible to the black horizontal line. As can be seen, increasing $s$ corresponds to faster minimization of CLIP error. However, even with comparable CLIP errors, visual effect might vary significantly. Large values of parameter $s$ are often associated with undesired artifacts. We find that values between [10, 30] are usually preferable.

Figure 8. Interpolation between the base subspace and the repurposed subspace where $s = 50$. As can be seen, undesired behavior occurring at repurposed subspace (e.g. blue skin Marge Simpson) cannot be mitigated by traversing shorter distances in test time. The choice of parameter $s$ is crucial in training time.

References


Figure 9. Subset 1/3 of generated images from a model expanded with 400 domains.
Figure 10. Subset 2/3 of generated images from a model expanded with 400 domains.
Figure 11. Subset 3/3 of generated images from a model expanded with 400 domains.


Figure 12. Composition of factors of variation introduced to a generator pretrained on FFHQ [12]. Following the format of Fig. 8.


Figure 13. Composition of factors of variation introduced to generators pretrained on LSUN Church [48] and SD-Elephant [20]. Following the format of Fig. 8.


[21] Yotam Nitzan, Kfir Aberman, Qiurui He, Orly Liba, Michal Yarom, Yossi Gandelsman, Inbar Mosseri, Yael Pritch, and


