Controlling BigGAN Image Generation with a Segmentation Network

Aman Jaiswal, Harpreet Singh Sodhi, Mohamed Muzamil H, Rajveen Singh Chandhok, Sageev Oore, and Chandramouli Shama Sastry

Dalhousie University, NS, Canada
{aman.jaiswal,harpreet,mohamed.muzamilh,rajveen,sageev,cssastry}@dal.ca

Abstract. GANS have been used for a variety of unconditional and conditional generation tasks; while unconditional generation involves learning and sampling from $P(X)$, conditional generation can be described as sampling from $P(X|f(X) = 1)$, where f is a binary indicator function. Most commonly studied conditional generation are class-conditional generation wherein f is a binary class-membership function. While class-conditional generation can be directly integrated into the training process, integrating more sophisticated indicator functions within the training is not as straightforward. In this work[1] , we consider the task of sampling from $P(X)$ such that the silhouette of (the subject of) X matches the silhouette of (the subject of) a given image; that is, we not only specify *what* to generate, but we also control *where* to put it: more generally, we allow a mask (this is actually another image) to control the silhouette of the object to be generated. The mask is itself the result of a segmentation system applied to a user-provided image. To achieve this, we use pre-trained BigGAN and SOTA segmentation models (e.g. DeepLabV3 and FCN) as follows: we first sample a random latent vector z from the Gaussian Prior of BigGAN and then iteratively modify the latent vector until the silhouettes of $X = G(z)$ and the reference image match. While the BigGAN is a class-conditional generative model trained on the 1000 classes of ImageNet, the segmentation models are trained on the 20 classes of the PASCAL VOC dataset; we choose the "Dog" and the "Cat" classes to demonstrate our controlled generation model.

Keywords: Generative model · Image Segmentation · Computational Creativity Tools

1 Introduction

GANS have been used for a variety of unconditional and conditional generation tasks; while unconditional generation involves learning and sampling from $P(X)$,

[1] This work was originally completed as a course project in the Deep Learning course, and has subsequently been revised and prepared for a conference submission (currently under review)

conditional generation can be described as sampling from $P(X|f(X) = 1)$, where f is a binary indicator function. Most commonly studied conditional generation are class-conditional generation wherein f is a binary class-membership function. While class-conditional generation can be directly integrated into the training process, integrating more sophisticated indicator functions within the training is not as straightforward. Specifically, in this work, we aim learn to replace the silhouette of a subject in an image (in our examples, an animal) with a different subject (e.g. a different animal) that still fits the exact same silhouette. Some generative models conditionally generate images by transforming vectors that lie in a large *latent* space. BigGAN, for example, has been trained to conditionally generate realistic images from any of the 1000 different categories of Imagenet, including various breeds of dogs and cats. While it is straightforward to sample from any of these categories; attributes like shape, size and posture cannot be directly manipulated to match our preference. These visible attributes are influenced by the choice of latent vector, but the nature of that influence is neither explicit, nor easily invertible, i.e. it is not clear how to choose a latent vector in order to achieve a desired visual attribute.

We introduce an iterative optimization-based approach to allow control over the silhouette of the image subject. We use a publicly-available pre-trained segmentation model to obtain a proxy for the silhouettes and the pre-trained BigGAN generator to conditionally generate our desired subject. We compute the differences in both silhouettes and optimize to iteratively produce images that can match silhouette of the given subject. This is done by (locally) optimizing over the latent-space of GAN until the euclidean distance between the segmentation maps is minimized. Figure 1 shows an example of our final system's output as it iterates to find an image whose silhouette matches that of the source image (8a).

1.1 Background

Our system depends crucially on two types of models: a GAN-based generator, and segmentation model. We discuss each of these.

Image generation. Generative adversarial networks (GANS) [6] use a neural network (G) to transform a latent vector z sampled from a prior distribution $p(z)$ to produce an output image $X = G(z)$. The generator network further comprises of intermediary layers $G_1..G_l$, where the first layer takes as input the latent vector to produce features tensors. The initial features are used by the next layer to produce higher abstraction of these features $y_i = G(y_{i-1})$. Lastly, The final layer is responsible for producing an output image $X = G_l(y_{l-1})$. Large scale class-conditional image synthesis [1] demonstrated that GANS could improve sample variety and fidelity from scaling up the number of parameters and batch size. BigGAN employs a shared class embedding c that is linearly projected to every layer and uses skip connections from the noise vector z to multiple layers of generator $y_i = G_i(y_{i-1}, z)$.This allows the latent space to directly influence features at different levels of hierarchy. This is done by splicing z into one chunk

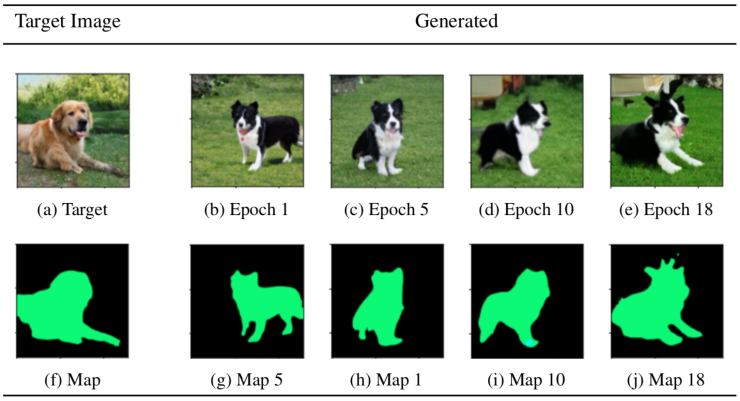

Fig. 1: Image (8a) is a provided source image. In this case, the source image happens to be itself a generated image (i.e. none of these images are photographs). Image (8b) shows a class-conditionally generated dog image for a random initial latent vector z, and (8c-8e) show the progression of images as we optimize z through the latent space (described in Section 3) to arrive at an image (8e) of a different dog from the original source, but whose *silhouette* matches that of the source image. These images were found using the *ensemble model* as described in Section 4. Images (8f)-(8j) show the corresponding segmentation maps.

per resolution and concatenating it with the shared class embedding c. They truncate the latent prior $N(0, I)$ during inference to improve sample quality. Although sampling from a truncated prior distribution during inference improves individual sample quality it also introduces undesirable saturation artifacts. In our framework, BigGAN provides a suitable generator for because of its ability to generate diverse high resolution samples including multiple species and breeds of various animals (useful for our example purposes). The architecture details of BigGAN are described in figure 2.

Image Segmentation. An important part of our framework is realised using *Semantic segmentation*. It can be explained by extending the idea of classification to the pixel level where an image is partitioned—or more accurately, the set of *pixels* of an image is partitioned—such that each pixel in a partition belongs the same class. Since the class of every pixel in the image is being predicted, this task is commonly referred as *dense prediction*. Earlier approaches have relied on primitive thresholding, clustering, edge-detection and graph-based methods for segmentation. In contrast, a majority of the work [18] [2] for segmentation in deep learning builds on convolution neural networks (or CCNs) which helps by learning increasingly abstract feature representations. However, this approach introduces challenges like reduced resolution which may impede dense prediction tasks, where detailed spatial information is desired. SOTA models like Deeplabv3 [3]use atrous convolution (also known as dilated convolutions) figure 3 to overcome this problem. For example, a kernel of size $K \times K$ with

$z \in \mathbb{R}^{160} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(20 + 128) \to 4 \times 4 \times 16ch$
ResBlock up $16ch \to 16ch$
ResBlock up $16ch \to 8ch$
ResBlock up $8ch \to 8ch$
ResBlock up $8ch \to 4ch$
Non-Local Block (64×64)
ResBlock up $4ch \to 2ch$
ResBlock up $2ch \to ch$
ResBlock up $ch \to ch$
BN, ReLU, 3×3 Conv $ch \to 3$
Tanh

(a) BigGAN 512 generator

$z \in \mathbb{R}^{140} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(20 + 128) \to 4 \times 4 \times 16ch$
ResBlock up $16ch \to 16ch$
ResBlock up $16ch \to 8ch$
ResBlock up $8ch \to 8ch$
ResBlock up $8ch \to 4ch$
ResBlock up $4ch \to 2ch$
Non-Local Block (128×128)
ResBlock up $2ch \to ch$
BN, ReLU, 3×3 Conv $ch \to 3$
Tanh

(b) BigGAN 256 generator

Fig. 2: *BigGAN generators architecture*

a dilation rate of N will cover $(N-1) * K \times (N-1) * K$ pixels for an expansion of $(N-1) \times (N-1)$. This allows to control the resolution of features while preserving the number of parameters.The pre-trained semantic segmentation model from deeplabv3 provides segmentation images of 20 classes in the PASCAL dataset including dogs and cats. This provides us the required segment masks to facilitate controlled generation from BigGAN.

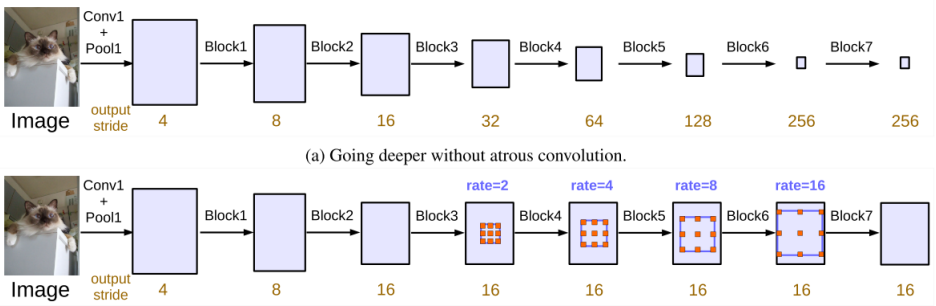

Fig. 3: Cascaded modules without and with atrous convolution ([3])

2 Related work

GAN frameworks [6],like BigGAN [1] and StyleGAN [10, 11] are powerful image synthesizers and have achieved impressive results in generation of variety of high quality images. Various improvements have been made to the original GAN model over the years, primarily to obtain higher quality images and more stable training, but most of those improved models still provide little direct control over the generated images other than selecting image classes or adjusting StyleGAN's style vector.

In studies like [5,9,15,16,20] there were attempts made to add control over the generated output images by focusing on supervised learning of latent directions. A few studies like [13, 17, 19] also provided useful control over spatial layout of the synthesized output images.

Our work focuses on exploring changes in the manifolds corresponding to the spatially localized region within the masked area of the image. We hope to discover smoothly varying sequences of latent vectors that lead to smooth transition of the generated "new subject" image (e.g. the new breed of dog) to exactly fit the mask corresponding to that of the target image (e.g. the silhouette of the dog in the provided source image).

A study by Yang et al [21] have explored similar results by applying a rectangular mask over features of the image like eye or mouth regions and learning a function that can be applied over the latent vector that allowed targeted control over the appearance of feature within the rectangular mask. Another study by Srinivas et al [8] shows that we can identify interpretable control over GAN generated image's pose, shape, facial and landscape attributes by applying principal component analysis (PCA) in latent space for StyleGAN, and feature space for BigGAN. Shen et al [16] propose a framework called InterFaceGAN, to identify the semantics encoded in the latent space of well-trained face synthesis models and then utilize them for semantic face editing. A paper by Nguyen-Phuo et al [14] proposes a novel method for the task of unsupervised learning of 3D representations from natural images. Their method enables direct manipulation of view, shape and appearance in generative image models. To generate new views of the same scene, transformations are applied to the learnt 3D features, and the results are visualised using a neural renderer that was jointly trained. Huang et al [7] propose a framework that decomposes the latent space of images into content space and style space and recombines the style spaces of different images to achieve style transfer.

Our attempt is to experiment with careful tuning of latent vector space in order to gain more control over the targeted portions of the generated image. We achieve this by changing the latent vector to fit the targeted image in the mask of the input image. An another study [12] proposes a solution to do face swap by combining neural networks with simple pre- and post-processing steps. We achieved subject-swapping for animals by pre-processing the input and by defining a loss function which takes input from both an image segmentation model and BigGAN model.

3 Model

A conditional GAN is a latent generative model that maps a point z in the latent space Z to an image $G(z)$ that follows a lifelike distribution R.The likelihood of $G(z) \sim R$ is influenced by the selection of z. Empirical evidence [4, 16] suggests that this mapping from $z \rightarrow G(z)$ is not always smooth and there are hidden but expressive transformations that remain to be explored. We propose a mask-guided image editing framework to swap a given subject in a given image (e.g. a dog) with another subject (e.g. a different dog!) using manifold transformation exploration. Our optimization framework requires four inputs: a source image X_s, a mask $M(X_s)$ (of the subject of interest) in the source image, a generated image X_g and its corresponding subject mask $M(X_g)$. Note that the user only provides a single source image X_s; the other image X_g is generated, and the masks of both images are obtained by the resnet-based semantic segmentation models. The framework $F(X_s, z_g)$,where X_s being the target image and z_g being the input latent vector for image to be generated optimises z to discover a meaningful transformation that can overlap the subject in the generated image X_g with that in X_s .The source image X_s can also come from another class of the generator. The optimization based exploration progresses using $L2$ loss between the source image segmentation map $M(X_s)$ and generated image segmentation map $M(X_g)$.

Our optimization framework is described in Figure 4. The segmentation model is used to get the segmentation maps for both the target image and the BigGAN generated image . Mean squared error is computed between the segmentation maps of the target image and the BigGAN generated image. The computed loss is then back propagated through the model to the input latent vector z of the generator. This vector, in turn, is optimised to minimize the MSE between these maps, and thus incrementally generate images that can fit within the segmentation map of the target image. We used the BigGAN generator due to both availability of pretrained parameters[2] and its ability to generate diverse samples.

4 Experimental Results

We initially tested this framework by performing small transformations, such as translations and rotation on a generated image, where we had access to the latent vector z_s used to generate the source image. This allows to assess model's ability to find the transformed vector from a good initialization point.This can be done by using the same BigGAN generated image as source and target image, where a known and controlled transformation has been applied to the source in order to generate the target. Figure 5 shows the results of this test.

The experiment in Appendix C (Figure 12) shows that the segmentation model struggles to segment the generated image when it is undergoing transi-

[2] https://tfhub.dev/s?network-architecture=BIGGAN,BIGGAN-deep&publisher=deepmind

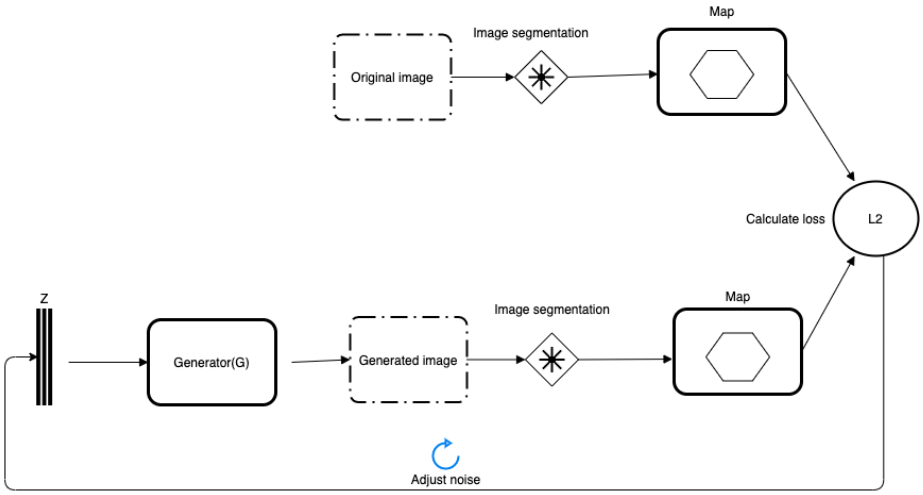

Fig. 4: Optimizing in Z: The top "row" of this figure stays fixed during optimization: given input image X_s is passed through a segmentation model to get a segmentation map, resulting in a mask $M(X_s)$. This is the source mask. In the bottom "row", the latent variable z is optimized using the $L2$ loss between the target map $M(X_s)$ and segmentation map $M(X_g)$ of the generated image X_g. The generated image, $X(g)$, is a itself generated based on the latent variable, i.e. $X_g = G(z)$. This allows us to incrementally update z until we are able to generate an image $G(z)$ such that its mask is very close to that of the source image, i.e. $M(G(z)) \approx M(X_s)$.

tions. The segmentation part of the model is regularised by adding another, second, segmentation model into the pipeline as shown in Figure 7. FCN ResNet101 is selected for supplementation because it has a global pixel-wise accuracy of 91.9% on COCO val2017 dataset and also shares the same architectural backbone as DeepLabv3 .

The average segmentation map is generated by computing the weighted average of the two maps (obtained by DeeplabV3 and FCN Resnet101). MSE loss is computed on the average maps of both the generated and target image. The model can benefit from averaging due to partial independent errors of the individual models. Appendix (B) contains further details about implementation. Another ensemble method which was also implemented but did not produce desirable results is discussed in Appendix A.

5 Discussion

We observed that when the generator did not yield a high fidelity initial image the model finds it difficult to converge and find a generation that can fit the target map. Also, we found that certain classes used by the generator seemed

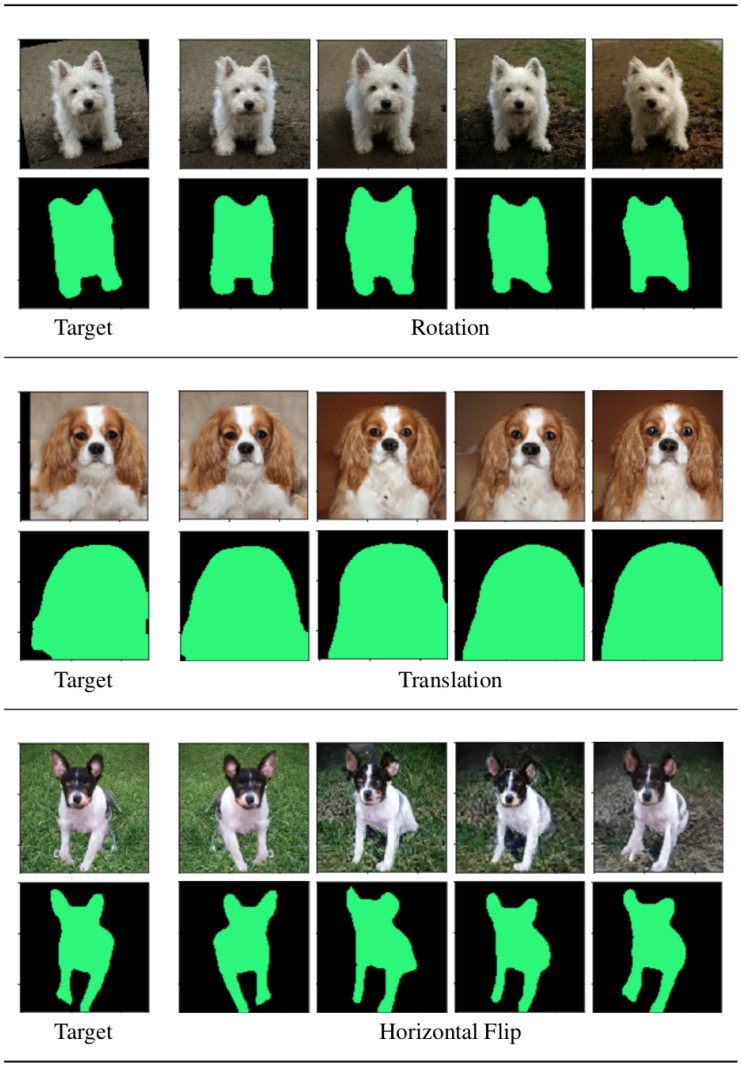

Fig. 5: Transformed generated image used as target. The target image is shown in 8a,The generated images are results from Epoch1,5,10,20 respectively.The rotation target is rotated 10 degrees to the left and the translation target is translated 10 pixels to the right

to allow better convergence than others, this may be attributed to the biases of the BigGAN generator. We experimented with both the original proposed pipeline and the extended ensemble version.The results shown in figure 8 are generated using a single segmentation model while computing MSE on the target and generated segmentation maps. The results in figure are generated using an ensemble of segmentation modules as illustrated in Figure 1.

Class Experiments The multi-task nature of the models used in the pipeline allows for generation of variety of animals. The segmentation models used are capable of segmenting birds and cats. Figure 6 demonstrate the ability of the model to fit different birds and cats. We can see in this example that that while the cat image had a very well-matched silhouette, the bird silhouette did not quite converge.

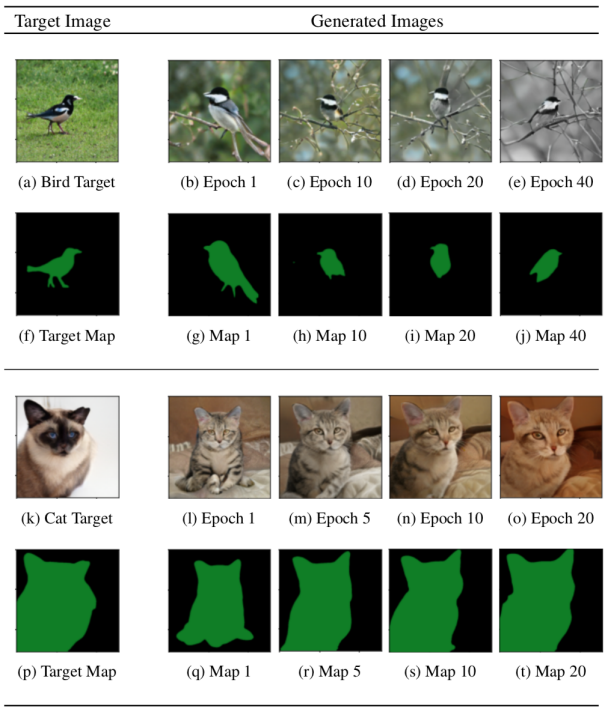

Fig. 6: *Different classes*, Figures (b)-(e) shows the the generated bird images with their corresponding segmentation maps from Figure (g)-(j). Figures (l)-(o) shows generated cat images with their corresponding segmentation maps from Figure (q)-(t).

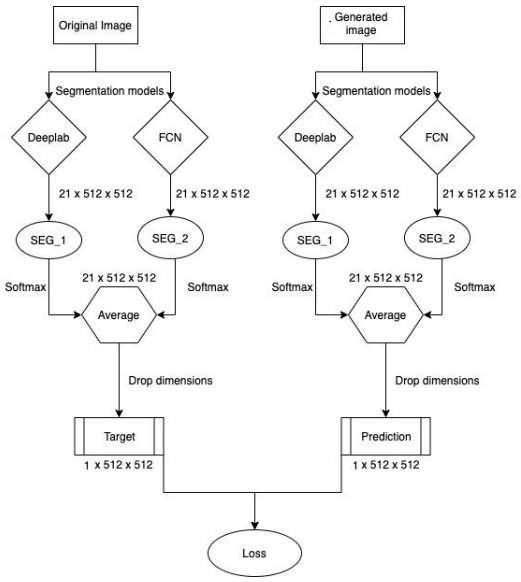

Fig. 7: *Ensemble of two segmentation models*

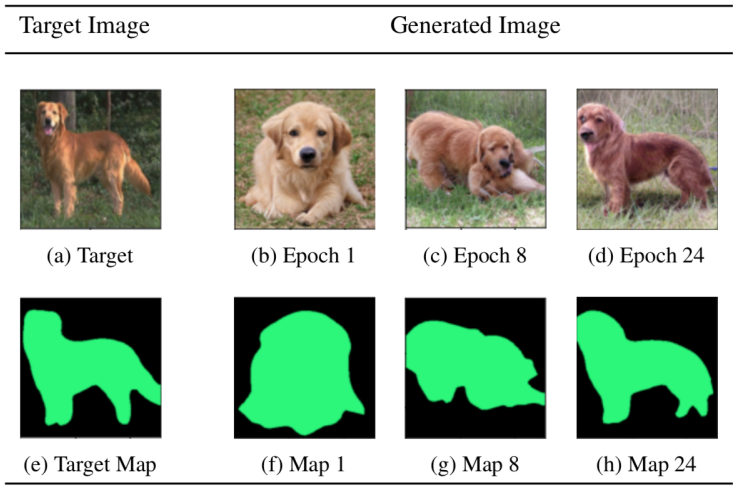

Fig. 8: *Shifting Dog Face*, target image is shown in (a), generated images are shown in (b), (c), (d) and their corresponding segmentation maps in (f), (g), (h). The face of the dog shifts from the right-side towards left-side .

6 Conclusion

In this paper, we were able to demonstrate that two independently trained modules when stacked together can achieve the task of subject swapping. Initial experiments showed poor segmentation of images undergoing transition, so the segmentation part of the model was regularised by adding an additional segmentation model in an ensemble fashion. Future work may include using a discriminator to further regularize the model to provide more gradient feedback.

References

1. Brock, A., Donahue, J., Simonyan, K.: Large Scale GAN Training for High Fidelity Natural Image Synthesis (sep 2018), http://arxiv.org/abs/1809.11096
2. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs (2016)
3. Chen, L., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. CoRR **abs/1706.05587** (2017), http://arxiv.org/abs/1706.05587
4. Creswell, A., Bharath, A.A.: Inverting the generator of a generative adversarial network (ii) (2018)
5. Goetschalckx, L., Andonian, A., Oliva, A., Isola, P.: Ganalyze: Toward visual definitions of cognitive image properties (2019)
6. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Networks (jun 2014), https://arxiv.org/abs/1406.2661
7. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation (2018)
8. Härkönen, E., Hertzmann, A., Lehtinen, J., Paris, S.: Ganspace: Discovering interpretable gan controls (2020)
9. Jahanian, A., Chai, L., Isola, P.: On the "steerability" of generative adversarial networks (2020)
10. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks (2019)
11. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan (2020)
12. Korshunova, I., Shi, W., Dambre, J., Theis, L.: Fast face-swap using convolutional neural networks (2017)
13. Kulkarni, T.D., Whitney, W., Kohli, P., Tenenbaum, J.B.: Deep convolutional inverse graphics network (2015)
14. Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., Yang, Y.L.: Hologan: Unsupervised learning of 3d representations from natural images (2019)
15. Plumerault, A., Borgne, H.L., Hudelot, C.: Controlling generative models with continuous factors of variations (2020)
16. Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of gans for semantic face editing (2020)
17. Singh, K.K., Ojha, U., Lee, Y.J.: Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery (2019)

18. Srinivas, S., Sarvadevabhatla, R.K., Mopuri, K.R., Prabhu, N., Kruthiventi, S.S.S., Babu, R.V.: A taxonomy of deep convolutional neural nets for computer vision. Frontiers in Robotics and AI **2** (Jan 2016). https://doi.org/10.3389/frobt.2015.00036, http://dx.doi.org/10.3389/frobt.2015.00036
19. Tran, L., Yin, X., Liu, X.: Disentangled representation learning gan for pose-invariant face recognition. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1283–1292 (2017). https://doi.org/10.1109/CVPR.2017.141
20. Yang, C., Shen, Y., Zhou, B.: Semantic hierarchy emerges in deep generative representations for scene synthesis (2020)
21. Yang, M., Rokeby, D., Snelgrove, X.: Mask-guided discovery of semantic manifolds in generative models (2021)

A Alternative Ensemble methods

The segmentation models used share similar architecture(Resnet101) and training dataset. Although the range of the logits vary from network to network, we could not find any evidence that computing an average across the logits produced by different segmentation modules should not necessarily produce good results. Therefore, we tried averaging the logits and then applying soft-max on the channel dimension before computing BCE Loss. The results of the average segmentation are shown in (Figure 9). We also tried a method where we average the losses. as illustrated in (Figure 10). This method did not work as well as the method illustrated in (Figure 7), The reason for this deviancy can be the BCE loss that we used while implementing this method.

B Implementation Details

We use a pytorch ported version[3] of the original model(As illustrated in Figure 7). The target and the generated image are used as inputs to two separate segmentation models. The pretrained segmentation models were taken from pytorch hub[4][5] . We use Adam optimzer with a learning rate of $1e-1$ and beta values of 0.5 - 0.99. The model is run for a maximum of 25 epochs. The segmentation models expects the RGB channel to have the corresponding Mean(μ) = $[0.485, 0.456, 0.406]$ and Variance (σ) = $[0.229, 0.224, 0.225]$ values, This is done explicitly for every generated image. *Mean squared error* is used for computing the loss over the "Dog" channel of the two segmentation maps. Weighted average with the ratio 0.6 : 0.4 is used for the segmentation models because the DeepLabv3 segmentation model works better than FCN resnet101 segmentation model.

[3] https://github.com/ivclab/BIGGAN-Generator-Pretrained-Pytorch
[4] https://pytorch.org/hub/pytorch_vision_fcn_resnet101/
[5] https://pytorch.org/hub/pytorch_vision_deeplabv3_resnet101/

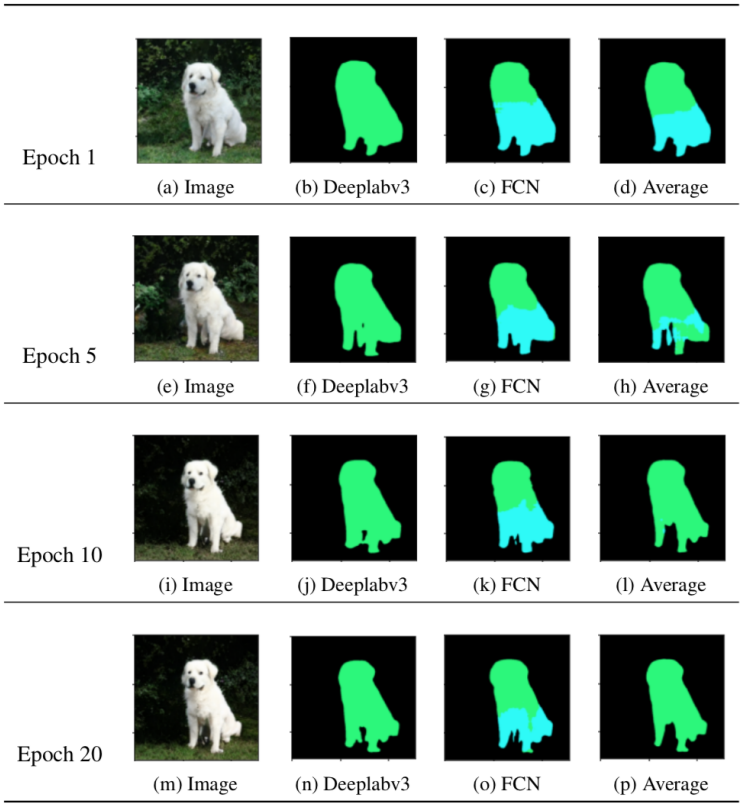

Fig. 9: Average over logits of two segmentation models, a) Deeplabv3 b) FCN ResNet101

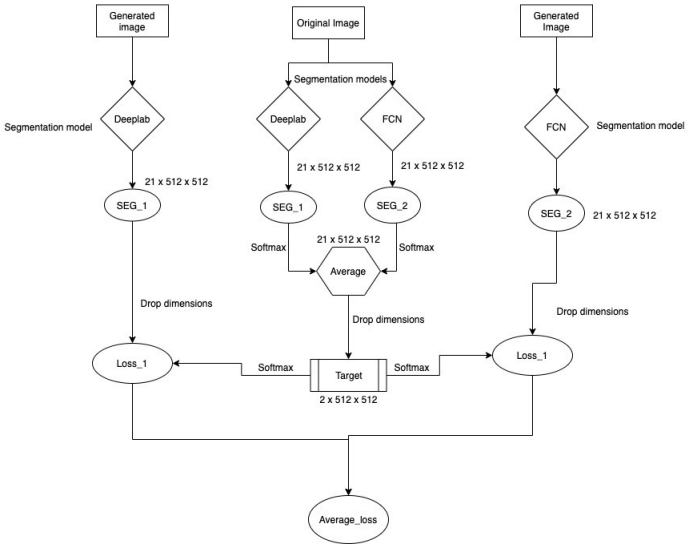

Fig. 10: *Average losses*

(a) Generated (b) Average Map

(c) Generated (d) Average Map

Fig. 11: *Background change* The model changes the background owing to inclusion of background channel in loss computation

C Loss and Channel Experiments

During training, we tried losses including cross-entropy, binary cross-entropy, soft cross-entropy, and mean squared error. The results of the segmentation models used contains 21 channels, where each channel outputs un-normalised probability values for pixels belonging to a particular class. Channel 0 is the "Background class". While performing the transformation experiments shown in Figure 5, we used binary cross-entropy loss. The channels used for computing the loss were the background channel and the "dog" channel. Figure 11 illustrates how the model tries to fit the background while reducing the loss. We found excluding the background channel and computing the mean squared error only on 'dog' channel works best.

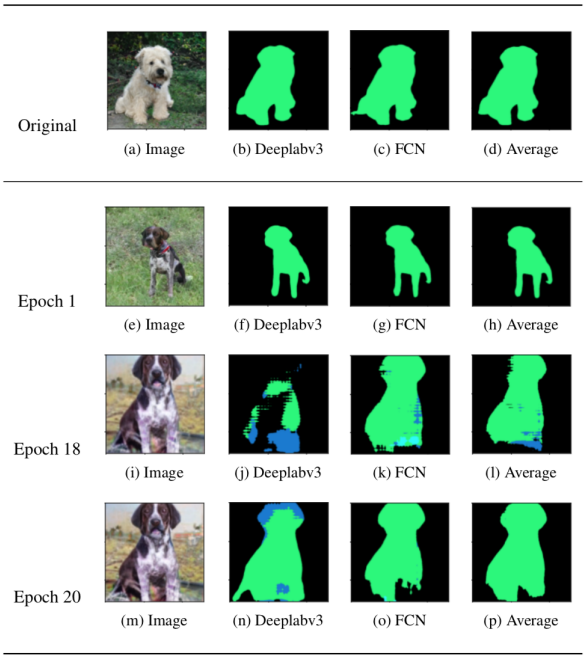

Fig. 12: *Ensemble Segmentation.* Image (j) shows poor segmentation by DeeplabV3 and Image (o) shows poor segmentation by FCN.