BranchGAN: Branched Generative Adversarial Networks for Scale-Disentangled Learning and Synthesis of Images

Zili Yi\textsuperscript{1,2,3}, Zhiqin Chen\textsuperscript{*2}, Hao Cai\textsuperscript{3}, Xin Huang\textsuperscript{3}, Minglun Gong \textsuperscript{3}, and Hao Zhang \textsuperscript{2}

\textsuperscript{1} Huawei Technologies Co. Ltd. \\
\textsuperscript{2} Simon Fraser University, Canada \\
\textsuperscript{3} Memorial University of Newfoundland, Canada

Abstract. We introduce BranchGAN, a novel training method that enables unconditioned generative adversarial networks (GANs) to learn image manifolds at multiple scales. The key novel feature of BranchGAN is that it is trained in multiple branches, progressively covering both the breadth and depth of the network, as resolutions of the training images increase to reveal finer-scale features. Specifically, each noise vector, as input to the generator network, is explicitly split into several sub-vectors, each corresponding to, and is trained to learn, image representations at a particular scale. During training, we progressively “defreeze” the sub-vectors, one at a time, as a new set of higher-resolution images is employed for training and more network layers are added. A consequence of such an explicit sub-vector designation is that we can directly manipulate and even combine latent (sub-vector) codes which model different feature scales. Experiments demonstrate the effectiveness of our training method in scale-disentangled learning of image manifolds and synthesis, without any extra labels and without compromising quality of the synthesized high-resolution images. We further demonstrate three applications enabled or improved by BranchGAN.

Keywords: Generative Adversarial Network; Image Synthesis; Image Representation Learning; Multi-Scale; Disentanglement

1 Introduction

Unconditioned GANs \cite{1} have been intensively studied as a means for unsupervised learning and data synthesis. Compared to their conditional counterparts \cite{2,3,4,5,6,7,8,9}, unconditioned GANs place less burden on the training data but are less steerable at the same time. In an unconditional GAN, a well-trained generator could synthesize novel data by sampling a random noise vector from the learned manifold as input and altering values “parameterizing” the dimensions of the manifold. However, this synthesis process is typically uncontrollable and counterintuitive, since we have little understanding how each manifold dimension impacts the synthesized output.
Fig. 1. Cross-scale image fusion by directly combining coarse-scale features in one image with finer-scale features from another. Please note that \( x^0 (x \in \{a, b\}) \) encodes image-wide structures and \( x^t (t \in \{1, 2, 3, 4\}) \) encodes increasingly fine-scale features. Given a pair of images, we compose new images by cross-combining coarse-scale structures and fine-scale features of the two, accomplishing expression transfer (a) and face swap (b).

For manifold learning of images or other visual forms, the notion of feature scales is of great importance. An ability to learn multi-scale or scale-invariant features often leads to a deeper and richer understanding of representations and distributions of images. In the last few years, scale-aware unconditioned GANs have been developed, i.e., StackGAN [9], LPGAN [10] and PGGAN [11], where correlated GANs are trained in a coarse-to-fine manner, using lower- and then higher-resolution images, with the goal of improving the quality of the final full-resolution images. However, factors which impact image features at various scales remain entangled in PGGAN networks. In StackGAN and LPGAN, factors (dropout layer or noisy perturbation) were added at various scales to learn scale-independent image features, though they are neither explicit nor controllable. Under the setting of conditional GANs, several recent works [12, 3], including DNA-GAN [13] and InfoGAN [14], aim to disentangle the latent codes which correspond to different image attributes. Namely, the image structures are disentangled by attributes rather than scales.

In this paper, we introduce a novel training method that enables unconditioned GANs to learn image manifolds in a “scale-disentangled” manner, aiming to improve the controllability of image synthesis and editing. The key novel feature of our learning paradigm is that each noise vector, as input to the generator, is explicitly split into a prescribed number of sub-vectors, i.e., 5 for learning 256\(^2\) images and 6 for 512\(^2\) images, where each sub-vector corresponds to, and is trained to learn, image representations at a particular scale. A consequence of such a sub-vector designation is that we can directly manipulate and even com-
We start with both the generator (G) and discriminator (D) having a low spatial resolution. During the first training period, we feed $z^0$ with random vectors of uniform distribution and $z^t$ ($t > 0$) with zero vectors, and make those linear layers corresponding to $z^t$ ($t > 0$) untrainable. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. Meanwhile, we “de-freeze” more $z$ vectors for training by feeding them with non-zero uniform-random vectors. This process is repeated until the target resolution is reached. During training, “branch suppression” (see Section 3) happens as $z^0$ has well encoded large-scale structures and will maintain its dominance in coarse-level encoding. When $z^1$ is de-frozen, it is suppressed in terms of coarse-level encoding but has the chance to encode finer-scale features. Figure 1 shows an example of cross-scale image fusion, where we intentionally synthesize an image by integrating the coarse-scale features of one image with finer-scale features of another.

At the high level, our learning method employs the standard GAN framework, which comes with an unconditioned generator and a discriminator, and follows the standard GAN training paradigm as described in [1, 15]. To achieve scale-disentangled learning, our network is trained progressively, bearing some similarity to Karras et al. [11]. However, instead of progressing only on network depth (adding network layers as the resolutions of training images increase), our network training also progresses over network width by progressively activating sub-vectors that correspond to different feature scales; see Figure 2 and more details in Section 3. As a result, we explicitly designate dimensions of the image manifold to different image scales, leading to scale-specific “training branches”. Hence, we refer to our network as BranchGAN.

By disentangling the image scales, BranchGAN enables multi-scale learning of image manifolds and more controllable image editing and synthesis, without
requiring extra labels. We tested our novel training method on several high-quality image datasets to verify its effectiveness in learning scale-disentangled image representations, compared with alternative GAN training schemes. We show that BranchGAN enables new applications of GANs to coarse-to-fine image synthesis and scale-aware image fusion. We further demonstrate improved performance of iGAN [16], an interactive image generation method, when the generator is trained by BranchGAN in place of DCGAN.

2 Related work

2.1 Multi-scale image representation

An inherent property of visual objects is that they only exist as meaningful entities over certain ranges of scales in an image. How to describe image structures at multiple scales remains an essential and challenging problem in image analysis, processing, compression, as well as image synthesis. Early methods for multi-scale image representing such as Discrete Fourier/Cosine Transforms (DFT/DCT) [17] and Discrete Wavelet Transformation (DWT) [18] are widely used in decomposing small-scale details and large-scale structures. In our paper, DFT is employed as a metric for scale disentanglement.

Another scale-independent representation of images is the layer activations of a well-trained Convolutional Neural Network (CNN) [19, 20, 21]. In a CNN, top activation layers roughly represent large-scale image structures such as objects and scenes, while bottom activations represent small-scale details such as edges, colors, or textures. Other than CNNs, stacked models such as Deep Belief Network (DBN) [22], Stacked AutoEncoders (SAE) [23, 24] or multi-scale sparse Coding [25] can also be utilized to retrieve multi-scale representations of images, though the effectiveness could be limited.

2.2 Coarse-to-fine image synthesis

Scale-aware image synthesis has been explored in StackGAN [6], LPGAN [10], and PGGAN [11], as we discussed in Section 1. The goal of these methods is to synthesize higher-quality images, rather than to learn multi-scale image manifolds. We extend the idea of progressive training from progressively adding layers to progressively growing both layers and branches. In addition, instead of training multiple GANs [6, 10], our method trains only one GAN.

2.3 Controllability in image synthesis

One of the most frequently adopted approaches to improving the controllability of image generation is conditional modeling. For example, conditional GANs or semi-conditional GANs can condition the image synthesis task on image attributes [3, 12, 13, 14], classes [4], input texts [5, 6], or images [7, 8, 9]. These methods either require extra labels or paired training images, or need strict inherent relations between the priors and the outputs. Our method, as a type of
Training sub-procedure at one scale level. After a new layer is added to the generator, we first only train the last layer of the generator while holding other layers untrainable (Stage I). After the last layer is well-trained, we then de-freeze all pre-trained layers (the branches in green) plus the newly-added branch for training (Stage II). To avoid “sudden shock” to the well-trained layers, we feed the newly de-frozen $z$ sub-vector with a random vector following uniform distribution $U(-\alpha, \alpha)$, where $\alpha$ increases smoothly from 0.0 to 1.0 throughout Stage II.

unconditional GAN, conditions image generation on random noise of uniform distribution and does not require any extra labels or priors.

For unconditional GANs, the method known as iGAN [16] provides a way for users to synthesize or manipulate realistic images in a more controllable way. In iGAN, users could add certain constraints on the appearance of desired images (i.e., draw edges, add color strokes or set an exemplar image) and the latent code is then optimized to satisfy these constraints. Nonetheless, a sophisticated optimization method is required, as gradient descent is particularly vulnerable to local minima. We demonstrate through experiments that the scale-disentangled latent spaces learned with our method can help iGAN avoid falling into local minima and hence boost its performance.

3 BranchGAN training

As shown in Figure[2], we start with training only the sub-vector corresponding to the coarsest level features, i.e., using the lowest-resolution images, while keeping the other sub-vectors “frozen”. Then we progressively “de-freeze” the sub-vectors,
one at a time, as a set of higher-resolution images is employed for training and more network layers are added. When training for a finer scale, the network weights learned from the previous coarser scales are used for initialization. Note that after training, these weights are often changed to adapt to the new training data.

What had motivated our branched GAN idea and what made it work effectively is a phenomenon that we observed during our experiments with multi-branch data generators and coined as “branch suppression”. Roughly speaking, we found that when multiple noise vectors, with their respective training branches are at play, GAN training typically results in one dominant branch while the other branches are either fully or partially suppressed. In other words, the already-trained weights (branches) will have priority in maintaining their role in encoding the image structures that are already encoded and suppress the other branches.

Branch suppression does happen to the proposed branched training method. When de-freezing one sub-vector during progressive training, branch suppression helps inhibit the ability of the newly defrozen branch in the network to encode coarser-scale structures, thus “encouraging” it to encode the finer-scale structures in the new set of higher-resolution training images. Note that the inhibition or suppression is not absolute; the network weights in previously trained branches are still altered.

BranchGAN progressively adds the depth (or layers) and breadth (or \( z \) sub-vectors) to the generator, and then start training with images of higher-resolution. During the process, branch suppression helps encourage the newly-added sub-vector to encode the finer-scale structures, as shown in Figure 2. At each scale, a two-staged sub-procedure is used to avoid sudden shock to already well-trained, smaller-resolution layers: see Figure 3. Note that all already-added layers of the discriminator are trainable throughout the sub-procedure. More details about hyper-parameters are available in the supplementary material.

The architecture of the generator is the same as shown in Figure 2. For 256 × 256 image generation, we use 5 \( z \) sub-vectors. The number of \( z \) sub-vectors is subject to change according to the resolution of output images. We use generator and discriminator networks that are mirror images of each other and always grow in synchrony, and use the standard non-saturated loss as in DCGAN [15] for training.

4 Results, evaluation, and applications

We have tested and evaluated BranchGAN on three datasets: church_outdoor from LSUN [20], celeba_hq [11], and car. The original car dataset has 800 × 600 pixel resolution. To speed up the training, we used downsampled versions of celeba_hq (256 × 256) and car (400 × 300). We trained our models on a GTX TITAN XP GPU, which took roughly 20 ~ 40 hours per (full) dataset.
Fig. 4. Effects on generated images for celeba_hq dataset by varying individual sub-vectors. We first initialize $z$ randomly, and then replace one of the sub-vectors $z_t$, $t = 0, \ldots, 4$, by $pI$, where $p = -0.8, -0.4, 0, 0.4$, or 0.8 and $I$ is the all-one vector of length $|z_t|$, while holding all the other sub-vectors fixed. Columns 1 to 5 show images generated by BranchGAN and the last column shows a variance image for the five generated images on the left, where lightness reflects pixel variance. From top to bottom, changing $z^t$ leads to smaller and smaller image variations, as reflected by intensity drop in the variance images. Sub-vector $z^0$ dominates the overall color, $z^1$ controls some facial features, while the rest bring minor changes near ear, mouth, and hair.

4.1 Evaluation of scale disentanglement

Qualitative evaluation. We wish to show how each designated sub-vector affects images generated via BranchGAN training. To this end, we vary the values of each sub-vector while holding the other sub-vectors fixed, as shown in Figures 4, 5, and 6. From visual examination and as detailed in the figure captions, we can observe that $z^0$ affects the output images most significantly as it mainly controls large-scale structures, which contrasts the effects of $z^4$ or $z^5$ as they are tied more to smaller-scale details. This suggests that scale disentanglement by splitting the training to progressively activated sub-vectors has been achieved.
Fig. 5. Effects on generated images for car dataset by varying individual sub-vectors. The output setting is the same as in Figure 4. The first row basically reflects the property of the car dataset: it contains left-views and right-views, but no front-views. GAN was not able to learn a smooth interpolation between the two views, resulting in a messy image in the middle of the first row. However, once $z^0$ is fixated with a view, the other sub-vectors can generate smooth interpolations. $z^1$ appears to alter the view angles slightly, $z^2$ impacts the front parts of the car, while changing the other sub-vectors influences more minor details.

That being said, it is also clear that we do not have direct control for feature localization or image semantics.

Quantitative evaluation. To examine how each dimension of the latent manifold space impacts appearance of the output images, we designed a metric to evaluate the variance of the output images when the latent vector is manipulated. The metric, which we refer to as variance by scale or VBS, denoted by $\mathcal{V}$, measures the variation of output images with respect to any sub-vector $z'$ of $z$, at a specific scale, as reflected by a frequency interval $[f_1, f_2]$. That is, $z'$ can correspond to a single dimension of $z$ or to one of the designated sub-vectors $z^t, t = 0, \ldots, 4$. 
Fig. 6. Effects on generated images for church-outdoor dataset by varying individual sub-vectors. The output settings is the same as in Figure 4. Similarly, as more clearly reflected in the variance images, the sub-vectors $z^t$, with $t$ from 0 to 4, appear to control higher-level to finer image details.

Specifically,

$$V_{f_2}^{f_1}(z') = \sum_{h,w,d} \mathbb{E}_{z' \sim U(-1,1), \overrightarrow{c} \leftarrow c, \overleftarrow{c}} (\text{DFT}_{f_1}^{f_2}(G(z))),$$

and

$$V_{f_3}^{f_1}(z') = V_{f_2}^{f_1}(z')/\mathbb{E}_{z' \subseteq z} V_{f_1}^{f_2}(z'),$$

where $\overrightarrow{z}$ is the set of dimensions of $z$ excluding $z'$, $\sigma_{z' \sim U(-1,0,1.0), \overrightarrow{c} \leftarrow c} f(z)$ refers to the deviation of the value of $f(z)$ when $z'$ follows the uniform distribution $U(-1.0,1.0)$ and $\overrightarrow{z}$ is fixed as a constant vector $c$. $G(z)$ is the output image of the generator $G$ given $z$. $h, w, d$ are the height, width, and depth of images (or layer activations). $\mathbb{E}(\cdot)$ is the expectation operator. In Eq. (1), $\text{DFT}_{f_1}^{f_2}(\cdot)$ refers to the discrete Fourier transform of an image, and $(f_1, f_2)$ is a frequency range. In order to avoid the impact of image size, VBS is further normalized by division over their expected values. Intuitively, a larger value of VBS implies a greater...
Fig. 7. Cross-scale image fusion. The notations and synthesis setup are the same as in Figure 1. From the first three columns, we can see that the sub-vector $x_1$ mainly controls the facial features while $x_2$ controls the face shape. By swapping $x_1$ and $x_2$, a face swap may be achieved. $x_3$ and $x_4$ have less significant impacts, such as lighting, shading, and minor changes in hair and ear. More such examples can be found in the supplementary material.

Impact of a manifold dimension (or a subset of manifold dimensions) on the output.

To examine the distributions of VBS at specific scales, we split the frequency domain into five ranges: $(0, 1/16)$, $(1/16, 1/8)$, $(1/8, 1/4)$, $(1/4, 1/2)$, and $(1/2, 1)$, which roughly correspond to increasingly fine image scales. We then visualize the VBS distributions for various GANs using histogram plots, as shown in Figure 8. To produce the histograms, we randomly sampled 10 $z$ vectors to generate 10 images using the trained model for the celeba_hq dataset. For each $z$ vector, which is of dimension 150, and for each dimension, we vary it while keeping all the other dimensions fixed. This results in a set of varied images, from which we compute the VBS value for the selected dimension. Overall, we collect
Fig. 8. Distributions of VBS over specific frequency ranges or image scales. The VBS of DCGAN [15], PGGAN [11] and InfoGAN [14] predominantly falls into the interval $(0.5, 1.5)$, whereas the VBS of BranchGAN spans over a wider range of $(0.1, 2.5)$.

Fig. 9. Plot of VBS values for sub-vectors $z^0, z^1, \ldots, z^4$ against frequencies. Peak VBS values indicate maximal impact. For example, sub-vector $z^0$ exhibits higher impact over the lowest frequency range, which corresponds to larger image scales.
10 \times 150 = 1,500 \text{ VBS values to form the histograms, for each GAN option and for each frequency range. As can be observed from Figure 8, the VBS values of BranchGAN exhibit a much greater variance (i.e., wider histogram) than those of traditional GANs, such as DCGAN [15], PGGAN [11], and InfoGAN [14]. Specifically, the VBS values of these traditional GANs at all scale levels mainly fall into the range of \([0.5, 1.5]\), implying that the corresponding image representations are more scale-entangled, in comparison to BranchGAN, whose VBS values vary over a larger interval \([0.1, 2.5]\).

Finally, to examine whether the image manifolds learned by BranchGAN are disentangled by the designated sub-vectors \(z^t, t = 0, \ldots, 4\), we show how the VBS of each \(z^t\) varies against frequencies in Figure 9. These VBS values were obtained in the same way as for the histogram plots in Figure 8. As we can observe, \(V(z^0)\) sees its peak value in the \((0, 1/16)\) range, implying that \(z^0\) mainly controls larger-scale structures of the generated images. The remaining sub-vectors \(z^1, z^2, \ldots, z^4\) show their peaks at frequency intervals which reflect their respective controls over increasingly finer image features or structures.

4.2 New applications

Coarse-to-fine image synthesis. We show that the scale disentanglement afforded by BranchGAN facilitates coarse-to-fine image synthesis. To this end, we developed a new interactive application; see Figure 10 (a). A user can select best-matching faces from randomly-generated ones displayed on the right panel. At the coarsest scale, the images are mapped from different \(z^0\) values with other sub-vectors set to zero. If the user is satisfied with a coarse-level image, then he/she can select it and move on to the next scale. Then, the value of \(z^0\) will be fixed and images mapped from different \(z^1\) values will be displayed for selection. As a result, the user can progressively improve the appearance of a synthesized face, as shown in Figure 10 (b).

Cross-scale image fusion. Scale disentanglement facilitates another new application: cross-scale image fusion, where latent codes representing different scales are joined to create hybrid images. Figures 1 and 7 show some examples of such image fusion, which are synthesized by integrating coarse-scale features of one image with fine-scale features of another. Through swapping sub-vectors representing different scales, our approach can achieve coarse-level fusion, such as face swap, as well as fine-level fusion, such as expression and face shape transfer.

4.3 Improving interactive image editing (iGAN)

We show how BranchGAN can improve the performance of iGAN [16], an interactive image editing tool. In the original paper, DCGAN [15] was adopted. We now compare different choices of GANs as replacement for DCGAN, including PGGAN [11], InfoGAN [14], and BranchGAN.

In the iGAN framework, a user makes interactive edits (i.e., scribbles, warping) to an existing image. The edits may be unprofessional and lead to various
Table 1. Average minimum optimization loss of iGAN using different GANs. DCGAN was used by the original iGAN.
other GANs indicates that BranchGAN generally performs better in fitting both coarse-level structures and fine-scale features.

5 Conclusion, limitation, and future work

We have introduced BranchGAN, a novel, progressive training procedure for unconditional GANs which enables multi-scale image manifold learning and manipulation. The key idea is to not only progressively increase network depth by adding layers, but also increase the network width by creating multiple, progressively activated training branches triggered by different sub-vectors of the network input. Each sub-vector corresponds to, and is trained to learn, image representations at a particular scale, leading to a scale-disentangled learning scheme. Experimental results on several well-known high-quality image datasets verify the effectiveness of our method in disentangling image manifolds by scales. We also demonstrated new and improved applications by GANs via BranchGAN training.

BranchGAN is scale-aware, but not feature-aware. This is a major limitation to progressive training using images at a selected set of resolutions. One reason is that not all image features are well represented in this selected set of training images. Another reason is that similar or repeated features in an image may not always be in the same scale, e.g., due to perspective projection. While such features are often manipulated as a collection during editing, they are difficult to learn using the current BranchGAN. In addition, the scale disentanglement afforded by our current approach is only a partial one, since adding a new training layer, which corresponds to a newly activated sub-vector, can still impact weights learned for the preceding sub-vectors. As a result, all learned weights may be correlated with image features across multiple scales. Overall, while the controllability enabled by BranchGAN training for image manipulation has been improved, it is still inherently limited.

In future work, we would like to extend BranchGAN to feature- or semantic-aware progressive training, where sub-vector designation can be based on more meaningful or more visually apparent image features; this would add more meaning to sub-vector manipulation for image editing and synthesis. We believe that the progressive training paradigm introduced by BranchGAN is a generic approach and can be tuned for different forms of disentanglement by adjusting the training targets. In addition, we shall explore potential values of scale-disentangled image manifolds in tasks such as image compression, filtering, and denoising. Finally, it is a curious question whether branch suppression exists in other “multi-branch” neural networks, such as ResNet [28], DenseNet [29], and capsule networks [30]. We are interested in whether this phenomenon may offer insights to the training of other convolutional and/or generative networks.

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS. (2014) 2672–2680
[2] Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv (2014)
[3] Yan, X., Yang, J., Sohn, K., Lee, H.: Attribute2image: Conditional image generation from visual attributes. In: ECCV, Springer (2016) 776–791
[4] Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. arXiv (2016)
[5] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. arXiv (2016)
[6] Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X., Metaxas, D.: StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: ICCV. (2017) 5907–5915
[7] Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. arXiv (2017)
[8] Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. arXiv (2017)
[9] Yi, Z., Zhang, H., Tan, P., Gong, M.: Dualgan: Unsupervised dual learning for image-to-image translation. arXiv (2017)
[10] Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using laplacian pyramid of adversarial networks. In: NIPS. (2015) 1486–1494
[11] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv (2017)
[12] Donahue, C., Lipton, Z.C., Balsubramani, A., McAuley, J.: Semantically decomposing the latent spaces of generative adversarial networks. In: ICLR. (2018)
[13] Xiao, T., Hong, J., Ma, J.: DNA-GAN: Learning disentangled representations from multi-attribute images. In: ICLR. (2018)
[14] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In: Advances in Neural Information Processing Systems. (2016) 2172–2180
[15] Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv (2015)
[16] Zhu, J.Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: ECCV, Springer (2016) 597–613
[17] Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. IEEE transactions on Computers 100(1) (1974) 90–93
[18] Shapiro, J.M.: Embedded image coding using zerotrees of wavelet coefficients. IEEE Transactions on signal processing 41(12) (1993) 3445–3462
[19] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11) (1998) 2278–2324
[20] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012) 1097–1105
[21] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv (2014)
[22] Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural computation 18(7) (2006) 1527–1554
[23] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR 11(Dec) (2010) 3371–3408
[24] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: ICML, ACM (2008) 1096–1103
[25] Sallee, P., Olshausen, B.A.: Learning sparse multiscale image representations. In: NIPS. (2003) 1351–1358
[26] Yu, F., Zhang, Y., Song, S., Seff, A., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv (2015)
[27] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Volume 1., IEEE (2005) 886–893
[28] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016) 770–778
[29] Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: CVPR. (2017)
[30] Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: ICANN, Springer (2011) 44–51

6 Appendix

6.1 Networks architecture and hyperparameters

Code for the models is available will be made available. The detailed information about the networks architecture of generators and discriminators are presented in Table 2, 3, 4, 5. The non-architecture hyper-parameters are listed in Table 6. Please note that lrelu is leaky relu layer.

| activation size | filter size |
|-----------------|-------------|
| input           | [30], [30], [30], [30], [30] | NA |
| concat          | [150] | NA |
| linear          | [32768] | [32768,150] |
| reshape         | [8,8,512] | NA |
| deconv+instanceNorm+lrelu | [16,16,256] | [5,5,512,256] |
| deconv+instanceNorm+lrelu | [32,32,128] | [5,5,256,128] |
| deconv+instanceNorm+lrelu | [64,64,64] | [5,5,128,64] |
| deconv+instanceNorm+lrelu | [128,128,64] | [5,5,64,64] |
| deconv+sigmoid (output) | [256,256,3] | [5,5,64,3] |

Table 2. Network architecture of the generator for 256 × 256 image synthesis.

6.2 Initialization of neural weights, “freeze” and “defreeze”

For the untrained linear or deconv/conv/linear layers, the filter weights are initialized with normally random numbers $N(\mu, \sigma)$ and biases are initialized with 0. For instance normalization layer, we initialize the scale with 1.0 and assign the center with 0.0.
### Table 3. Network architecture of the discriminator for $256 \times 256$ image synthesis.

| Layer                      | Activation size | Filter size |
|----------------------------|-----------------|-------------|
| Input                      | [256,256,3]     | NA          |
| Deconv+InstanceNorm+lrelu  | [128,128,64]    | [5,5,3,64]  |
| Deconv+InstanceNorm+lrelu  | [64,64,64]      | [5,5,64,64] |
| Deconv+InstanceNorm+lrelu  | [32,32,128]     | [5,5,64,128]|
| Deconv+InstanceNorm+lrelu  | [16,16,256]     | [5,5,128,256]|
| Deconv+InstanceNorm+lrelu  | [8,8,512]       | [5,5,256,512]|
| Reshape                    | [32768]         | NA          |
| Linear                     | [1]             | [32768,1]   |

### Table 4. Network architecture of the generator for $400 \times 300$ image synthesis.

| Layer                      | Activation size | Filter size |
|----------------------------|-----------------|-------------|
| Input                      | [30], [30],     | NA          |
| Concat                     | 150             | NA          |
| Linear                     | 17920           | 17920,150   |
| Reshape                    | 5,7,512         | NA          |
| Deconv+InstanceNorm+lrelu  | 10,13,256       | [5,5,512,256]|
| Deconv+InstanceNorm+lrelu  | 19,25,128       | [5,5,256,128]|
| Deconv+InstanceNorm+lrelu  | 37,50,64        | [5,5,128,64]|
| Deconv+InstanceNorm+lrelu  | 75,100,64       | [5,5,64,64] |
| Deconv+InstanceNorm+lrelu  | 150,200,64      | [5,5,64,64] |
| Deconv+sigmoid (output)    | 300,400,3       | [5,5,64,3]  |
Table 5. Network architecture of the discriminator for 400 × 300 image synthesis.

| name                                | activation size | filter size |
|-------------------------------------|-----------------|-------------|
| input                               | [300,400,3]     | NA          |
| deconv+instanceNorm+lrelu           | [150,200,64]    | [5,5,3,64]  |
| deconv+instanceNorm+lrelu           | [75,100,64]     | [5,5,64,64] |
| deconv+instanceNorm+lrelu           | [38,50,64]      | [5,5,64,64] |
| deconv+instanceNorm+lrelu           | [19,25,128]     | [5,5,64,128]| |
| deconv+instanceNorm+lrelu           | [10,13,256]     | [5,5,128,256]| |
| deconv+instanceNorm+lrelu           | [5,7,512]       | [5,5,256,512]| |
| deconv+instanceNorm+lrelu           | [17920]         | NA          |
| linear                              | [1]             | [17920,1]   |

Table 6. Non-architecture training hyperparameters.

| name                        | value                          |
|-----------------------------|--------------------------------|
| Optimizer                   | AdamOptimizer                  |
| learning rate               | 0.0002                         |
| beta1                       | 0.5                            |
| beta2                       | 0.999                          |
| #sub-vector                 | 5 for 256 × 256 images         |
|                             | 6 for 512 × 512 or 400 × 300 images |
| #epoch/scale                | 20 for 256 × 256 images        |
|                             | 12 for 512 × 512 or 400 × 300 images |
| #batch/epoch                | subject to dataset size and batch size |
|                             | we use full dataset for each epoch |
| batch size                  | 20 for 256 × 256 images        |
|                             | 12 for 512 × 512 or 400 × 300 images |

We “freeze” certain branches (or weights) by feeding the corresponding z vector with 0. Here we intend to explain the reason why feeding zero vector makes the corresponding weights untrainable.

Therefore, the activations of linear layer when fed with 0 are given by $f(0) = \theta 0 + 0 \equiv 0$, where $\theta$ are the linear weights. The activations of conv/deconv layer are given by $f(0) = \theta \otimes 0 + 0 \equiv 0$ (or $\theta \otimes 0 \equiv 0$ after concatenation), where $\theta$ are filter weights. So we have the gradients $\nabla f_{\theta}(0) \equiv 0$. For instance normalization layer and leaky relu layer, $g(0) \equiv \text{lerelu}(0 - 0 \cdot 1.0 + 0.0) \equiv 0$. We have the gradients $g_{\beta}(0) \equiv 0$, where $\beta$ is the scale.

In this way, the branches (or weights) could be “frozen” when fed with 0. To “defreeze” these branches (or weights), simply feed them with non-zero vectors.

6.3 Branch suppression

We observed “branch suppression” in all kinds of multi-branch generators as shown in Figure 14, among which some are fully suppressed, some are partially
suppressed. In “branch suppression”, the already-trained weights (branches) will have priority in maintaining their role in encoding the image structures that are already well encoded and suppress the other branches. To explain it in more details, we present a few examples of branch suppression in Figure 14.

6.4 More image editing results with BranchGAN and iGAN

Figure 16 shows the outputs of BranchGAN on celeba_hq 512 × 512 dataset. Figures 15 shows more results of cross-scale image fusion. Figures 17, 19, 18, 20 and 21 show more results of iGAN using our multi-scale image manifold as the latent codes.

6.5 Consistency of VBS with human perception

We are not aware of universally accepted metrics to assess variation of outputs “by scale”, as Frechet Inception Distance and Inception Score do not. We proposed VBS as objective metrics for variation-by-scale. To assess consistency of the metric with human perception, we conducted a user study. We selected 48 pairs of result images (18 for each dataset) that were produced by controlling the value of different latent vectors of BranchGAN. We hired 20 Turkers to rate each pair in terms of level of variation. In the test, three options with elaborate explanations were shown to the Turkers: (a) large-scale variation; (b) median-scale variation; or (c) small-scale variation. The label with the most votes is treated as ground-truth. Then we compare the human-labeled results with those estimated by VBS (the scale level with highest score is used) and compute the percentages of agreements is 85.4%. The agreement rates of VBS and human perception are quite high from this preliminary study (significantly better than random), serving as an initial validation. Further explorations are certainly warranted.

6.6 Experiments with LAPGAN [10]

LAPGAN [10] is very similar to BranchGAN in terms of coarse-to-fine image synthesis and adding noise at multiple scale level, though the noise is added through dropout layer, which is neither controllable nor explicit. We did re-implement LAPGAN and attempt to add noise vectors explicitly as inputs to generators at each step. The results showed that the noise vectors are not responsible for any variation of the residual images. The reason could be that by using strictly paired training data, the upsampled conditioning image would deterministically generate the residual image.
Fig. 10. Coarse-to-fine image synthesis. (a): GUI of the application. (b): Sequences of images selected in a coarse-to-fine manner. Bounding boxes with the same color highlight the changes made by the user at each step: yellow box → thinner face, red box → less dimple, and blue box → red lip.
Fig. 11. Overview of modified iGAN workflow. As in [16], the encoder that projects an image onto a manifold needs to be trained in advance. Once the user makes an edit to the image, the edited image is mapped to a latent code in the manifold space, which is assigned as the initial value ($z_0$) of the latent vector. Then the latent vector $z$ is optimized to minimize the objective in Eq. (2).

Fig. 12. GUI of the modified iGAN tool. The main window includes an edit zone (left) and a display zone (right). The edit zone provides various tools and a canvas to help edit the color map and mask or produce the edge map. The display zone shows the result generated by iGAN based on the edits. See video in the supplemental material for more details.
Fig. 13. *Comparison of iGAN results when using different GAN manifolds as the latent space.* Original images were edited by users in different ways: (a) no edit, (b) face lightened, (c) hair erased, (d) adding edge map. Notably, none of the results perfectly fit the edited images, as patterns do exist in images generated by the same GAN model. Other than this, we observe that in (a) and (b), the head poses rendered by BranchGAN and PGGAN fit the inputs better. In terms of smaller-scale image features, BranchGAN generally performs better than other GANs, which could be observed in regions highlighted in red bounding boxes.
Fig. 14. Two examples of branch suppression in GANs. In these examples, we employ the training loss and discriminator of dcgan [15]. Here we change the architecture of the generator a bit by conditioning image generation on split \( z \) vectors \((z^t, t \in \{1, 2, 3\})\). In the upper row, the left branch is already well trained for image generation, and the middle and right branches are initialized randomly (see more details about the initialization in the supplementary material). Then we train the GAN by following the standard GAN training procedure as in [15]. After the training converges, the left branch dominates the output while the other are fully suppressed, as seen from the variance image on the right (see Fig. 4 for the meaning of variance image). In the lower row, the generator architecture is the same as traditional GAN except that the \( z \) vector is split. We train the left branch till converging, then de-freeze the middle branch for training till converging, and finally the right branch. Note that the number of training steps for each stage are equal and the pre-trained weights (or branches) are not frozen even after new branches are de-frozen. As a result, the middle branch is slightly suppressed and the right branch is severely suppressed as seen from the rightmost variance images.
Fig. 15. Results of \textit{Cross-scale image fusion}. The notations and synthesis setup are the same as in Figure 7.
Fig. 16. Effects on generated images for celeba_hq 512 × 512 dataset by varying individual sub-vectors. The output setting is similar as in Figure 4 except that there is one more sub-vector $z^5$ and each row is generated independently. From top to bottom, changing $z^t$ ($t \in \{0, 1, 2, 3, 4, 5\}$) leads to smaller and smaller image variations, as reflected by intensity drop in the variance images. Similar to Figure 4, sub-vector $z^0$ dominates the overall color, $z^1$ controls some facial features and hair features, while the rest bring minor changes near ear, mouth, and hair.

Fig. 17. The edge maps and color maps drawn by users and the corresponding image generation results with improved iGAN.
Fig. 18. Edits by users and the corresponding results generated by improved iGAN. (a) face erased. (b) face slimmed. (c) mouth replaced with a patch from another image. (d) hair darkened. (e) mouth closed. (f) hair turned brown. (g) hair turned brown, eye shadowed, and lips reddened. (h) face whitened and lips reddened.
Fig. 19. Face image (512x512) generation and editing results with improved iGAN. (a)-(b), results based on edge maps. (c)-(d), results based on masked color maps. (e)-(h), image editing results.
Fig. 20. Car image generation and editing results with improved iGAN: (a-b) results based on edge maps; (c-d) results based on masked color maps; (e-h) manipulation of existing images, including erasing license plate (e), changing body color (f & h), and adding extra edge map (g).
Fig. 21. Church image generation and editing results with improved iGAN: (a-b) results based on edge maps; (c-d) results based on masked color maps; and (e-h) image editing results.