Understanding Feature Transfer in Generative Adversarial Network Generators

Matthew Denton
Auburn University
mtd0020@auburn.edu

Joshua Gaston
Auburn University
jcg0031@auburn.edu

James Smith
Auburn University
jss0036@auburn.edu

Abstract

Transfer learning is commonly applied to CNN training as a means to achieve rapid network convergence and boost generalization ability. While great success has been found in these applications, transfer learning has yet to be applied to image generation tasks. The State-of-the-Art image generation networks, Generative Adversarial Networks, produce the most visually appealing images but suffer from long training times and convergence issues. We demonstrate that transferring generator layers previously trained on independent datasets can speed convergence regardless of the transfer scheme applied. Additionally, our experiments have transformative implications as an alternative approach to initializing networks for small datasets.

1 Introduction

Generative Adversarial Networks (GANs) [1], the State-of-the-Art (SotA) deep learning method for image generation, are distinguished by the ability to produce visually pleasing results. Compared to competing networks such as Variational Autoencoders (VAEs) [2] and PixelCNNs [3], which try to model an explicit density function, GANs use a Game-Theory approach in which a generator network learns a transformation from a latent vector into an image distribution without directly parameterizing the image distribution.

Even though GANs have produced the most visually pleasing images, these networks have proven difficult to train due to issues with finding a generator-discriminator Nash-equilibrium [4, 5]. Recently, Spectral Normalization Generative Adversarial Networks (SNGANs) have demonstrated the ability to converge to unique and visually pleasing solutions without introducing complicated hyperparameters or significant additional parameters to learn [6]. Given the success of these stable networks, higher-level examinations of GANs can now be completed with greater impact.

The purpose of this work is to investigate feature transfer in the SotA SNGAN. This work approaches the problem of image generation generalization by applying transfer learning to SNGAN generator networks. Yosinki et al. first investigated the generality versus specificity of neurons in Convolutional Neural Network (CNN) layers and demonstrated that transfer learning (i.e., training a network on a large, different dataset and then fine-tuning deep layers on a desired training set) not only outperforms random weight initialization but also boosts generalization even after training the network on the new dataset [7]. In our work, a similar approach is applied to GANS to increase network performance and convergence by first training networks on a large dataset and then fine-tuning deep layers to a unseen task. Specifically, this paper makes the following contributions:

1. We characterize the specificity of layers in generator networks in GANs based on the criterion given by [7].
2. We compare random weights to transfer weights on a variety of image generation task using both frozen and fine-tuned weights. The transfer weights are trained on the Imagenet dataset

[8] and the tasks vary in similarity to the transfer set. We show that there is a trade-off between performance and number of layers transferred, and consider factors such as training convergence and size of the task dataset in characterizing the success of different transfer methods (number of weights and frozen vs. fine-tuning).

3. We investigate feature transfer as a relation to both distance from the latent vector layer and the generated image. Specifically, we begin transferring layers from the latent subspace towards the image distribution and then repeat our experiments in reverse. Surprisingly, we find that rapid convergence and achievable inception score is independent of these two schemes.

2 Review

We begin by discussing the importance of visualizing features in CNNs and the extension to GANs. Next, we review feature transfer with respect to CNN classifiers. Finally, we discuss related works where features of GANs have been successfully transferred for a different purpose.

2.1 Visualizing Generator Features

Advances in deep learning have been strengthened by the ability to visualize features learned within deep, hidden layers. Yosinski et al. provide an in-depth discussion on feature visualization in [9]. In convolutional neural networks, it has been found that layers close to the input image learn simple feature whereas layers close to the classification decision learn high-level, complex representations. Work done in [10] presents multifaceted feature visualization which allow us to synthetically generate various features learned by a neuron.

For our work, it is important to understand how features are learned for generator networks. Nguyen et al. extend the concept of activation maximization to image generation in [11, 12]. Radford et al. characterize GAN features in [13] and show that certain learned features can be "turned-off", such as removing windows from images of generated rooms. However, there remains to be work that characterizes specificity of GAN generator layers as a function of location. For example, it is unknown whether layers closer to the latent space or image space learn high-level representations or basic features. Our work provides a means to make these characterizations by applying transfer learning to GAN generators.

2.2 Feature Transfer in Convolutional Neural Networks

Transfer learning was presented for CNNs by Yosinski et al. in [7]. Besides providing a means to boost generalization abilities and training convergence for tasks with small training data, this work demonstrated the specificity of layers in CNNs in terms of placement in the network. The authors presented experiments in which ImageNet is randomly split into two datasets, *A* and *B*. The baseline CNN is trained on *B* with randomly initialized layers while the experimental transfer networks are initialized with a combination of layers transferred from a network trained on *A* and randomly generated layers. Layers are transferred in terms of distance from the input layer. For the first network, only the layer immediately following the input layer is transfered. Additional networks transferring the remaining networks in sequential order are provided with the final network having all layers transferred.

The networks were trained and evaluated on *B* and experiments demonstrated that layers closest to the image layer are more generalized and can thus be transferred to other tasks whereas layers closest to the decision layer are specialized to a specific task and must be retrained. Our work provides similar insight into the functionality of GAN generator networks by transferring generator layers in a similar manner.

2.3 Feature Transfer in Generative Adversarial Networks

GANs provide a unique image generation technique in that they are comprised of two networks, a generator and a discriminator. GANs learn a transformation from a latent vector, $\mathbf{z}$, into an image distribution, $p_{data}(\mathbf{x})$, by simultaneously training a generator, G, that produces $\mathbf{x_{gen}} = \mathbf{G}(\mathbf{z}, \boldsymbol{\theta}^{(\mathbf{G})})$

to fool a discriminator, $D(\mathbf{x_{gen}}, \mathbf{x})$, whose job is to distinguish fake images $\mathbf{x_{gen}}$ from real images $\mathbf{x}$. This work focuses on feature representation in the generator network.

To the best of our knowledge, we are the first to directly transfer weights of generator networks to a new task. However, other works have re-purposed layers from both generators and discriminators and should therefore be mention. The work of [13] transfers the discriminator of a GAN for successful supervised learning applications. In [14], Huang et al. present stacked GANs which uses stacked pre-trained encoders to guide the training of generator networks. This work learns a generator as an inverted encoder, empirically demonstrating the transferability of generator features. Adversarial feature learning is discussed in [15] which maps data from images back into the latent subspace, thus demonstrating that feature representations are directly learned in GANs. Finally, [16] combines a VAE and GAN by sharing a generator network, demonstrating the flexibility of the generator to be used for multiple purposes.

3 Approach and Model Architectures

Our design approach is modeled off that of [7] but with a few discrepancies. Similar to the prior work, we randomly split our training data into two datasets, *A* and *B*. However, since GANs have difficulties in converging to meaningful images when using a large dataset, we use only 20% of the ImageNet dataset for each split (meaning we use 40% of the total available images in each experiment). We include instances where transferred layers are both unfrozen (i.e., transferred layers will be fine-tuned with a smaller learning rate than randomly initialized weights) and frozen (i.e., transfered layers will not be trained but still backpropagate a gradient to not interfere with training the remaining network).

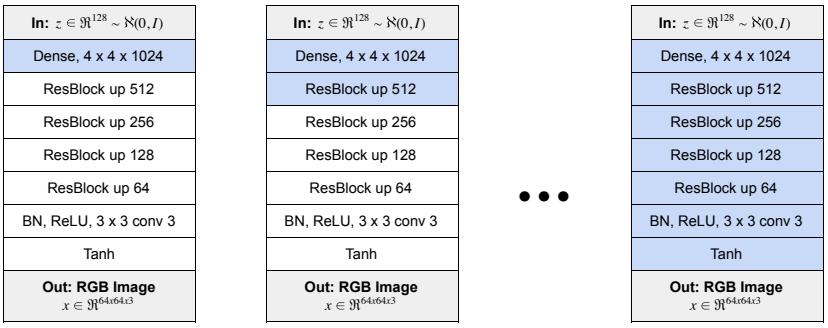
The discrepancies in our approach are a result of image generation being an unsupervised learning task rather than a supervised learning task. The primary difference is that generalizing to unseen data is not relevant for our task. We have two criterion of which to measure success: highest inception score achieved during training and convergence to the baseline inception score. We define the baseline inception score as the highest achieved inception score from the baseline network, a GAN trained and evaluated on *B* with randomly initialized layers. The transfer networks are defined as GANs evaluated on *B* that contain a combination of transferred layers from a network trained on *A* and randomly initialized layers. This work only looks at transferring layers in the generator network. A successful transfer learning network would converge to the baseline score in fewer iterations than the baseline network and achieve an even higher inception score during training. We will specifically quantify three instances for each training scenario:

1. Inception score before any training has taken place
2. Iteration at which inception score matches the baseline score (or a failure to do so)
3. Highest inception score produced during training

Transfer learning for CNNs is found to be most successful when transferring early layers (i.e., layers closest to the input layer) because these layers have been characterized by low specificity. But, as we discussed earlier, feature specificity is less known in GANs. It is for this reason that we set up our experiments in two separate transfer scenarios: forward and backward. Forward transfer, demonstrated in Figure 1, begins by transferring the layer closest to the input layer, and continues transferring layers sequentially towards the final layer until the entire network is transferred. Notice that this is the same method used for transfer learning in CNNs for image classification. Backward transfer, demonstrated in Figure 2, is the intuitive reversed extension of forward transfer. We begin by transferring the layer closest to the output layers and continue transferring layers sequentially backward until the entire network is transfered. The purpose of these separate scenarios is to not only ensure we provide the most effective manner for transferring GAN layers but to also be the first to characterize specificity in GAN layers in this manner.

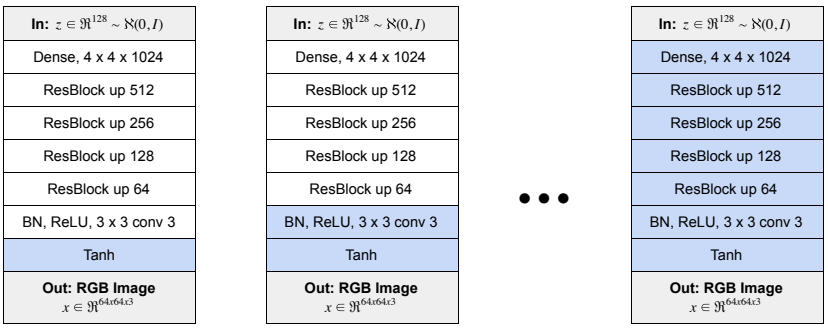
4 Empirical Validation

We performed two sets of experiments. The first set of experiments looks at transferring generator layers in the forward direction starting after the latent space, $\mathbf{z}$. The second set of experiments transfers generator layers in the backward direction starting from the layer before the image generated space, $\mathbf{x_{gen}}$. We performed each set in both the frozen and unfrozen scenario.



(a) 1 layer (b) 2 layers (c) All layers

Figure 1: Forward layer transfer scheme starting with a single layer transferred (a) and ending with all layers transfered (c). Transferred layers are highlighted in blue.



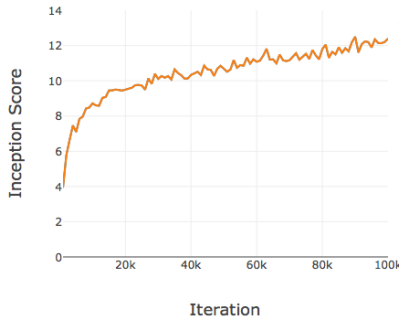
(a) 1 layer (b) 2 layers (c) All layers

Figure 2: Backward layer transfer scheme starting with a single layer transferred (a) and ending with all layers transfered (c). Transferred layers are highlighted in blue.

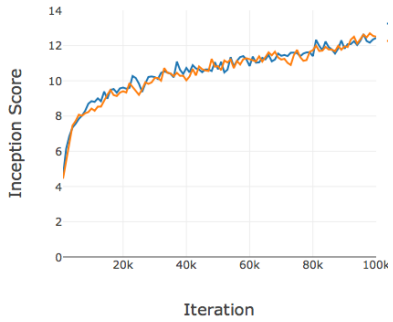
As mentioned earlier, we implemented our experiments using the SotA SNGAN [6], which is built on top of the Chainer framework [17]. The SNGAN model architecture is primarily composed of Resblocks [18] with a fully connected layer before the Resblocks and a convolutional layer with $tanh$ activation after. Return to Figures 1 and 2 for our model architecture and experiment design.

Our experiments produced interesting results both reinforcing expectations while introducing unexpected characterizations of transfer GANs. Training curves from each transfer scenario are given in Figures 3–4 while inception score vs number of layers transferred using the forward and backward schemes is given in Figures 5–6. Finally, generated images for two scenarios, the baseline and and highest transfer rate, are given in Figures 7. The following conclusions are made by looking at these results:

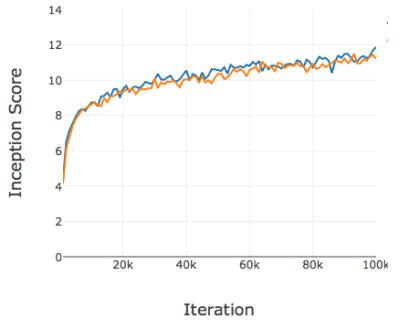
1. Weight transfer in GAN generator networks significantly decreases the iterations required for network convergence on inception score. As the number of layers transferred increases, the initial inception score sequentially increases for both the forward scheme and the backward scheme. This suggests that transferring entire GANs is ideal for rapid convergence, at least for the instance where the image distributions are of equal size.
2. Final inception score is not significantly increased by weight transfer and can, in fact, decrease the achievable inception score for a given network. This implies that it is advantageous to limit the layers that are transferred, which is consistent by the work in [7]. However, it is not evident that either the forward or backward method outperforms the other. These results were surprising; we expected to find that, similar to CNNs, the convolutional layers closest to the image layer would transfer the best. Because of this, it could be practical to transfer the weights closest to the latent space and, in the frozen scenario, reduce the number



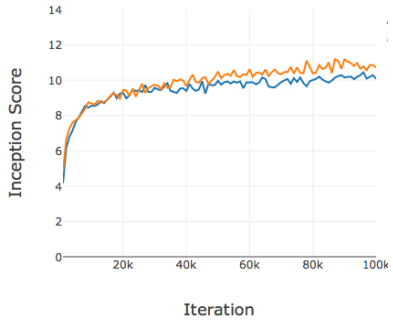
(a) No weights transfered (baseline)



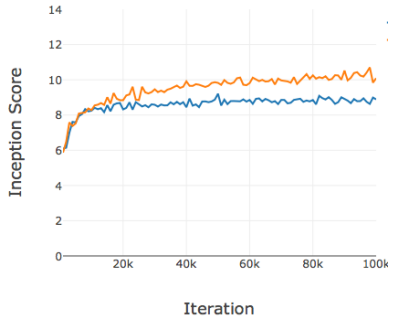
(b) 1 layer transfered



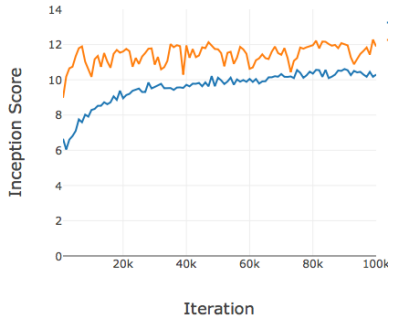
(c) 2 layers transfered



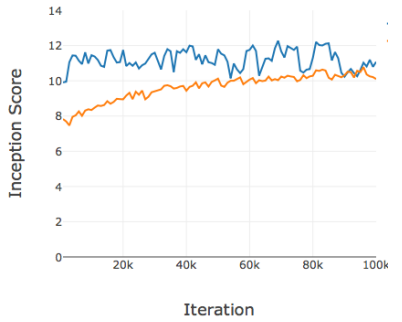
(d) 3 layers transfered



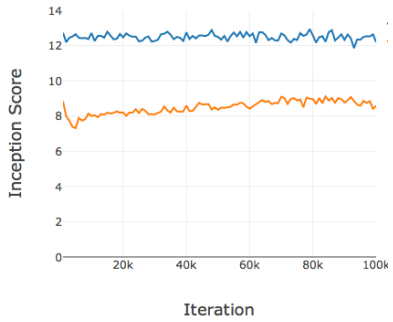
(e) 4 layers transfered



(f) 5 layers transfered

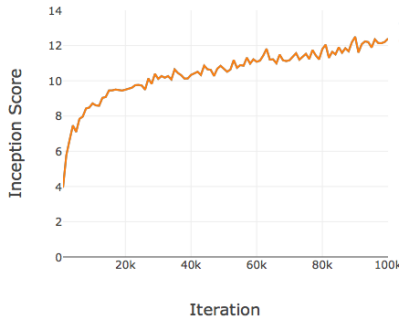


(g) 6 layers transfered



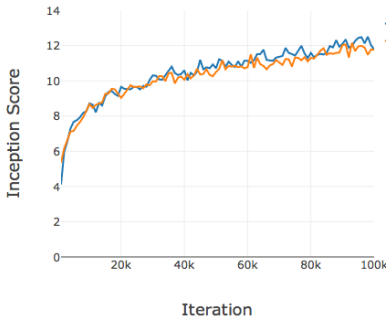
(h) 7 layers transfered (entire network)

Figure 3: Inception score vs. training iterations for forward transfer learning with frozen (blue) and unfrozen (orange) layers.

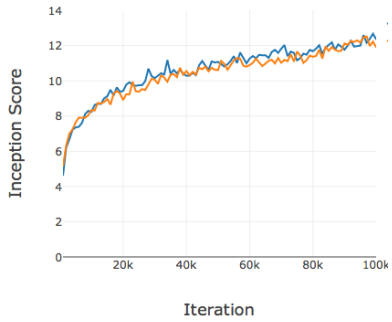


(a) Forward transfer

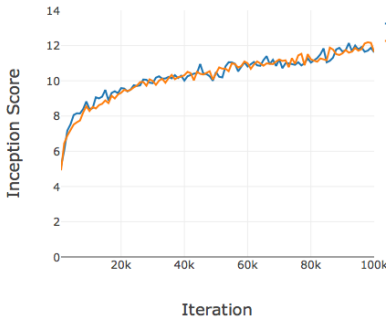
(a) No weights transfered (baseline)



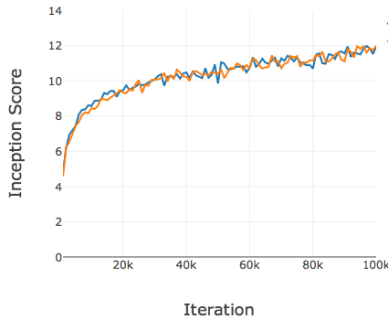
(b) 1 layer transfered



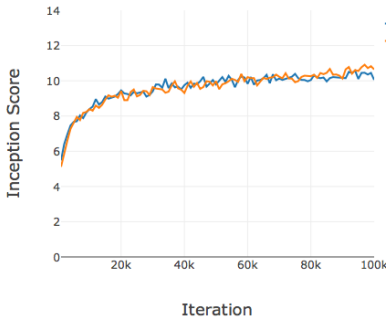
(c) 2 layers transfered



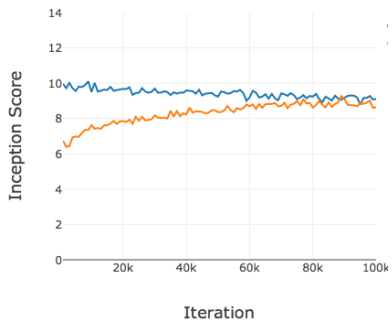
(d) 3 layers transfered



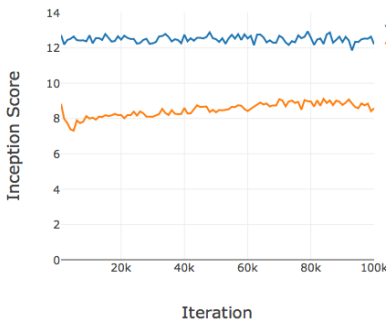
(e) 4 layers transfered



(f) 5 layers transfered

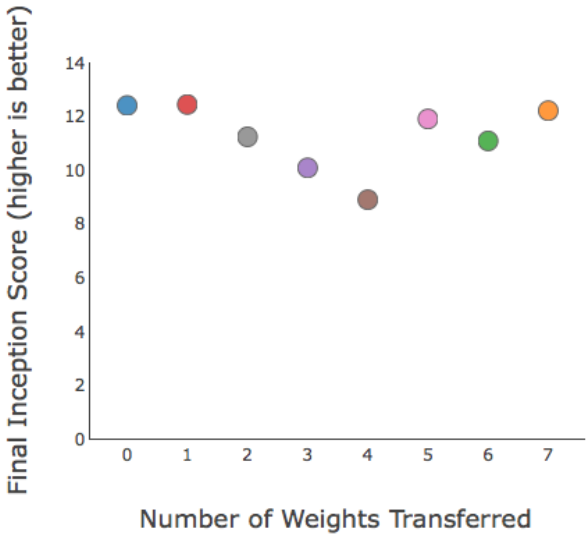


(g) 6 layers transfered

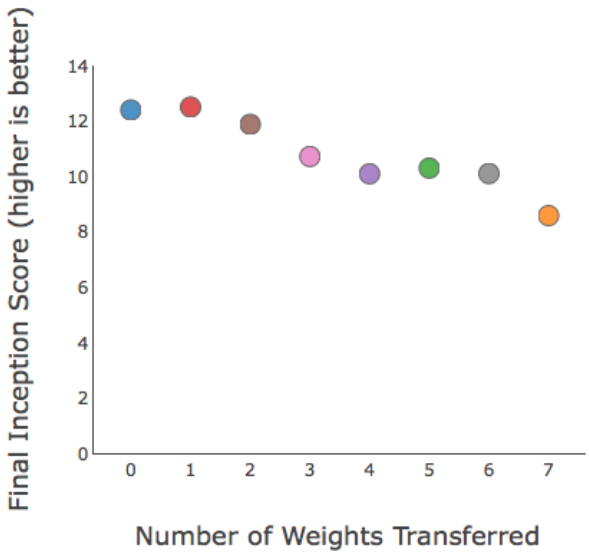


(h) 7 layers transfered (entire network)

Figure 4: Inception score vs. training iterations for backward transfer learning with frozen (blue) and unfrozen (orange) layers.

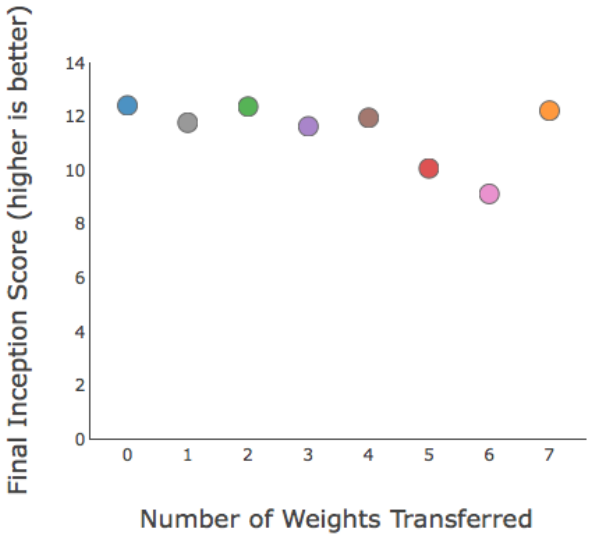


(a) Forward transfer – frozen

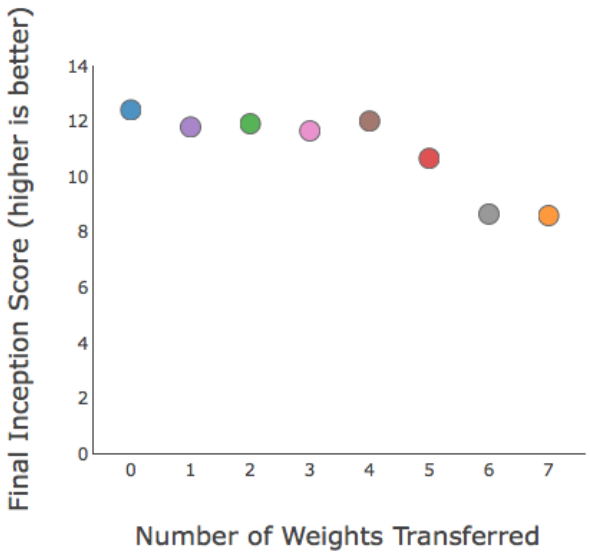


(b) Forward transfer – unfrozen

Figure 5: Final inception score vs. number of layers transferred for forward transfer learning with (a) frozen layers and (b) unfrozen layers



(a) Backward transfer – frozen



(b) Backward transfer – unfrozen

Figure 6: Final inception score vs. number of layers transferred for backward transfer learning with (a) frozen layers and (b) unfrozen layers

(a) Baseline initial

(b) Transferred initial

(a) Baseline final

(b) Transferred final

Figure 7: Final generated images for (a) initial baseline GAN, (b) initial transferred GAN, (c) final baseline GAN, and (d) final transferred GAN. Transferred network is four frozen layers transferred using forward scheme.

of computations involved in the weight update process by not back propagating the gradient throughout the entire network.

3. When only a few layers are transferred, there seems to be no significant difference between the frozen and unfrozen scenarios. However, as the last few layers are transferred (resulting in near or full generator transfer), the frozen weights seem to greatly outperform the unfrozen weights. This is unexpected and invites deeper analysis. Although not presented in this paper, the authors analyzed the generator and discriminator loss and observed that the fine-tuned weights of the unfrozen generator actually reduce the training loss more than the frozen generator. This is expected and consistent with transfer learning in CNNs. Therefore, the issue seems to be that inception score and training loss are not correlated in this particular instance. Specially, the frozen network continues to perform well because it has learned $p_{data}(\mathbf{x})$ from the original dataset, A, which is a large, natural image dataset similar to B. Therefore, we suspect that further experiments characterizing the effects of transfer learning between dissimilar image distributions will favor fine-tuning scenarios where several layers are transferred and favor freezing scenarios where only a few layers are transferred. It is important to observe that the significance of the described further experiments rely on the characterizations made in this paper and are therefore not included in this first-of-its-kind work.

5 Conclusion

Our work demonstrated the potential to achieve rapid inception score convergence in SotA GAN networks by transferring layers in the generator networks from GANs trained on a separate training dataset. We showed that, although we can improve training convergence, no overall increase in achievable inception score is gained (compared to increased generalization abilities in weight transfer found in CNNs). Finally, we found no significant difference between transferring layers starting from the latent space and starting from the image space. Future work will include baseline experiments of transferring weights trained on *B* to *B* to ensure the networks are specifically gaining from transferring knowledge from a uniquely different dataset. Additionally, we will look at transferring weights to distinctly smaller and dissimilar datasets to show benefits of transfer learning to tasks where either a small number of training images are available or where training images are of a unique type.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661. 1

[2] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, December 2013. arXiv: 1312.6114. 1

[3] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. *arXiv:1606.05328 [cs]*, June 2016. arXiv: 1606.05328. 1

[4] I. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160 [cs]*, December 2016. arXiv: 1701.00160. 1

[5] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. *arXiv:1606.03498 [cs]*, June 2016. arXiv: 1606.03498. 1

[6] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 1, 4

[7] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *arXiv:1411.1792 [cs]*, November 2014. arXiv: 1411.1792. 1, 2, 3, 4

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2

[9] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding Neural Networks Through Deep Visualization. *arXiv:1506.06579 [cs]*, June 2015. arXiv: 1506.06579. 2

[10] A. Nguyen, J. Yosinski, and J. Clune. Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks. *arXiv:1602.03616 [cs]*, February 2016. arXiv: 1602.03616. 2

[11] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space. *arXiv:1612.00005 [cs]*, November 2016. arXiv: 1612.00005. 2

[12] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *arXiv:1605.09304 [cs]*, May 2016. arXiv: 1605.09304. 2

[13] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434 [cs]*, November 2015. arXiv: 1511.06434. 2, 3

[14] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked Generative Adversarial Networks. *arXiv:1612.04357 [cs, stat]*, December 2016. arXiv: 1612.04357. 3

[15] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial Feature Learning. *arXiv:1605.09782 [cs, stat]*, May 2016. arXiv: 1605.09782. 3

[16] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv:1512.09300 [cs, stat]*, December 2015. arXiv: 1512.09300. 3

[17] chainer: A flexible framework of neural networks for deep learning, April 2018. original-date: 2015-06-05T05:50:37Z. 4

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 4