GAN You Do the GAN GAN?

Joseph Suarez

Abstract

Generative Adversarial Networks (GANs) have become a dominant class of generative models. In recent years, GAN variants have yielded especially impressive results in the synthesis of a variety of forms of data. Examples include compelling natural and artistic images, textures, musical sequences, and 3D object files. However, one obvious synthesis candidate is missing. In this work, we answer one of deep learning's most pressing questions: GAN you do the GAN GAN? That is, is it possible to train a GAN to model a distribution of GANs? We release the full source code for this project under the MIT license.[1]

1. Introduction, Background, Related Work

GANs (Goodfellow et al., 2014) have become perhaps the single most prevalent class of generative models in recent years, spawning hundreds of variants and an entire subfield of surrounding work. While they are notoriously difficult to train and often suffer from mode collapse, under the right conditions, GANs have been shown to generate compelling artificial data distributions and are perhaps best known for realistic image synthesis (Zhu et al., 2017).

Our objective is to apply GANs to a different class of data: GANs themselves. Instead of using a GAN to model images, we use a GAN to model a distribution of GANs that model images. We refer to this architecture over architectures as a GAN-GAN. We do not consider GAN-GAN-GANs or GAN-GAN-GAN-GANs in this work. While these are straightforward to implement, they would require an exponential parameter budget (at least as formulated in the present work) which we lack the hardware to support.

Hypernetworks (Ha et al., 2016) and extensions thereof (Deutsch, 2018; Suarez, 2017) have also explored the concept of using one network to generate the weights of another. However, to our knowledge, all such settings typically learn both the architecture and the meta-architecture end-to-end. In contrast, we are interested in whether it is possible to directly learn a distribution over architectures within the
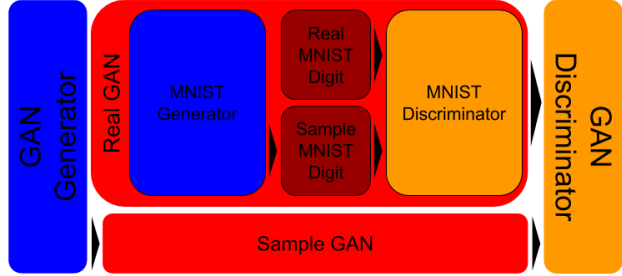
---

[1] **Full source code:**
https://github.com/jsuarez5341/gan-you-do-the-gan-gan

Figure 1. GAN-GAN: a GAN trained on a dataset of GANs

standard setting of generative modeling. We treat GANs themselves as examples and learn a GAN-GAN by training on small set of trained GANs.

This paper is a joke; however, the results are in fact real. Interpretability is a key problem in deep learning, especially in models to be deployed in real world systems. Generative modeling over networks could serve as a useful tool for visualization and analysis of network decisions and results.

2. Methods

GANs formulate a two player game between networks. Formally, GANs defines a Generator $\mathcal{G}$ and a Discriminator $\mathcal{D}$. The Discriminator models the probability $P(\mathcal{G}|x)$ that a given example x is fake. The Generator maximizes the probability $P(\mathcal{D}(\mathcal{G}(z)))$ (z is sampled noise) that the Discriminator will output an incorrect prediction. A GAN-GAN is simply a GAN trained on a dataset of GAN weights.

Algorithm 1 GAN-GAN Training. We first train a set of GANs and save snapshots of the parameters each epoch. We then train a GAN-GAN (a GAN over GANS) by treating each snapshot as an individual training example.

---

for GAN Index = 1...#Networks **do**
　Initialize an MNIST GAN
　for Epoch = 1...#Epochs **do**
　　Train the MNIST GAN for one epoch
　　Save a snapshot of the GAN parameters
　end for
end for
Load all snapshots of all GANs into a dataset with #Networks $\times$ #Epochs examples
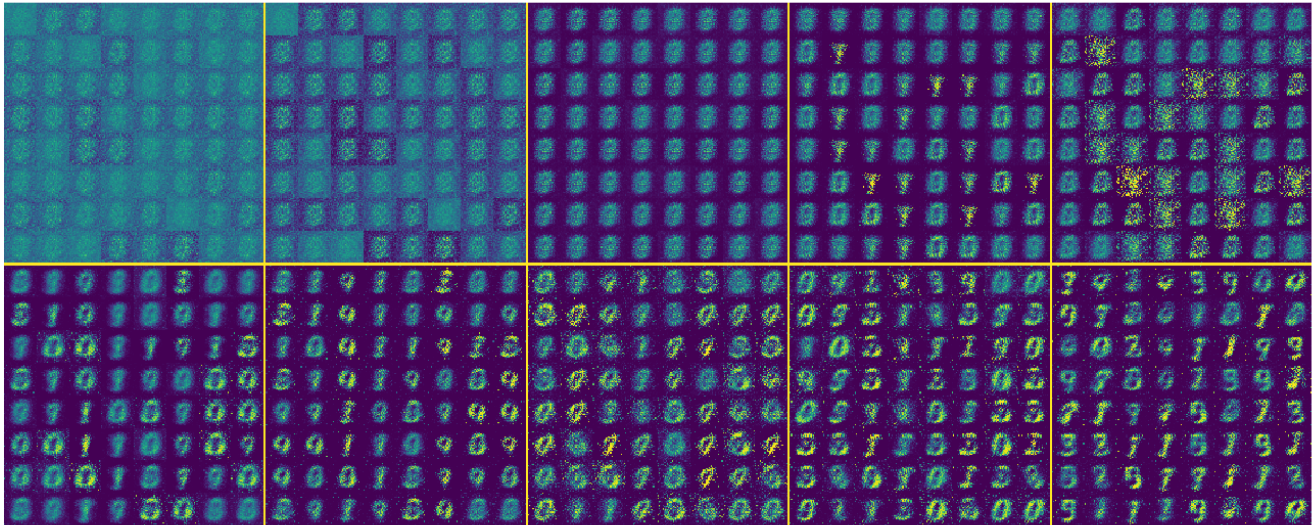Train a GAN over the dataset of GAN snapshots

---

Figure 2. Example samples from the training of an MNIST GAN (top-bottom left-right: epochs 1, 2, 10, 25, 27, 30, 32, 35, 40, 49)
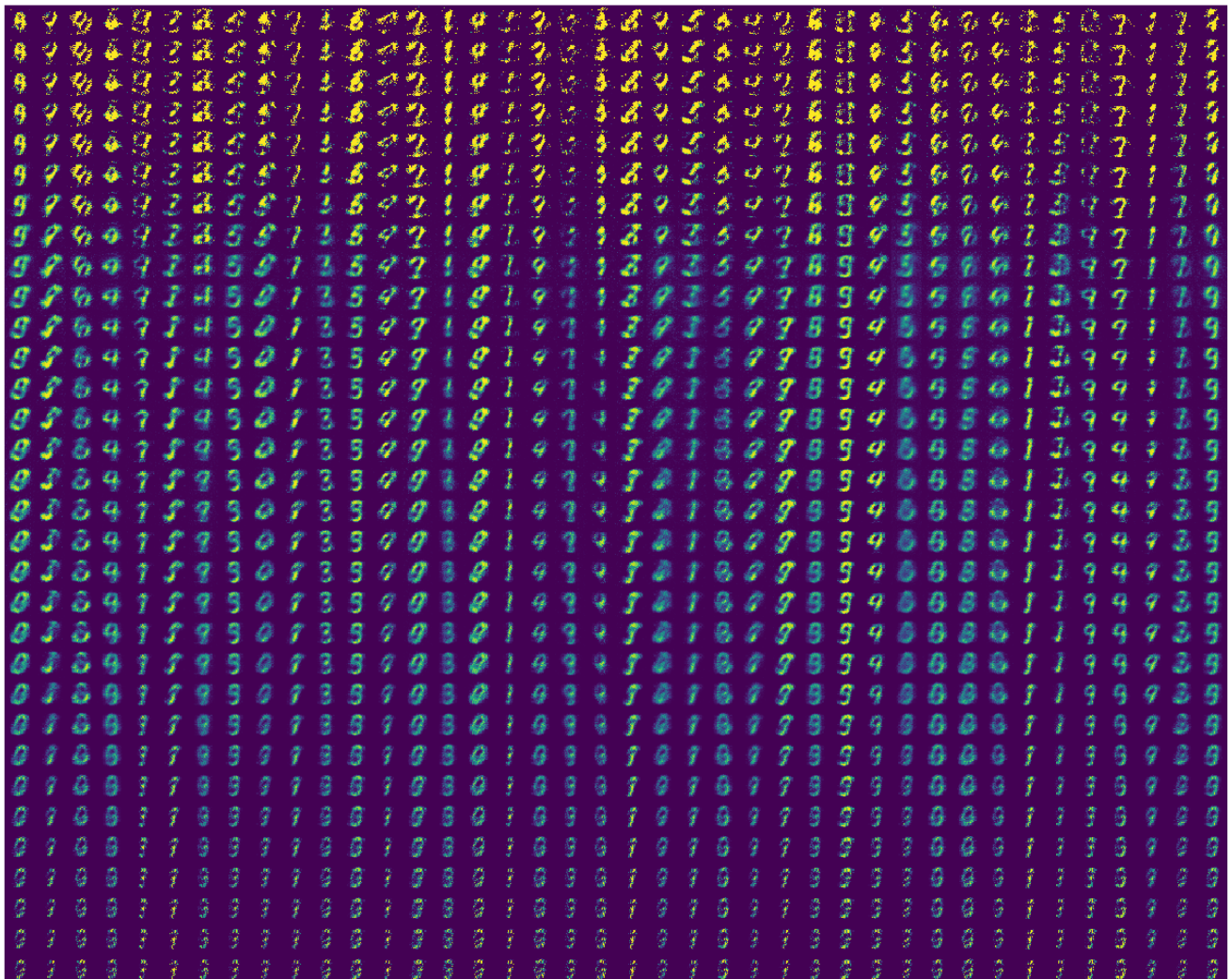

Figure 3. Image samples from GANs sampled from the trained GAN-GAN. Rows correspond to GANs linearly sampled from 1D GAN-GAN latent space in the interval (-2, 2). Columns correspond to a particular noise vector input to all GANs.

3. Experiments and Discussion

GAN Architecture   The generator and discriminator are three layer (input-hidden-output) fully connected neural networks with hidden dimension 64. We use Leaky ReLU(Xu et al., 2015) activations with 0.2 negative slope after the input and hidden layers. We use *tanh* for the generator output (dimensionality $28 \times 28 = 784$ to match MNIST) and *sigmoid* for the discriminator output (dimensionality 1). The generator samples from latent dimension 64.

GAN-GAN Architecture   The GAN-GAN generator and discriminator have the same layer and activation structure as the MNIST GAN. The input dimensionality is 113745, which is equal to the dimensionality of the GAN parameter vector. We found that using a smaller hidden dimension for the discriminator (8) than the generator (64) helped to stabilize training. We use latent dimension 1 for the GAN-GAN in order to enable visualizations. Results improve with a larger latent space.

Training   We use Adam(Kingma & Ba, 2014) for all networks. The learning rate is fixed to 0.0002; all other parameters are PyTorch defaults. We use batch sizes 128 and 32 for MNIST and the GAN-GAN, respectively. As described in Algorithm 1, we train 35 MNIST GANs for 100 epochs each, saving snapshots of the weights at each epoch. We train the GAN-GAN for 250 epochs using these 3500 snapshots as training examples.

Results   In order to evaluate the performance of the GAN-GAN, we first linearly sample 32 GANs from the 1-dimensional latent space of the GAN-GAN. We then fix 40 noise samples with the same dimensionality as the GAN latent space. Finally, we sample 40 images from each GAN using the fixed noise samples. Fig. 3 shows the results. The GAN-GAN produces neatly ordered GAN samples according to image quality. Surprisingly, the GAN-GAN actually exhibits better performance than the trained GANs. Given that high quality GANs appear more real than low quality GANs, it is possible that the GAN-GAN learned to bootstrap its own performance by implicitly combining snapshots (Liu et al., 2018) in latent space, in effect producing a smoothed history over training.

Discussion

While the GAN-GAN samples are of high quality with respect to the MNIST GAN samples, the MNIST GAN samples themselves are of fairly low quality (Fig. 2). As the weight dimensionality of each GAN is equal to the input dimensionality of the GAN-GAN, we were constrained to use a very small network for the MNIST GAN. One obvious solution is to use a more parameter efficient network architecture such as DCGAN (Radford et al., 2015). While we

were able to obtain significantly better image samples using a slightly modified DCGAN with under 30k parameters, we found it more difficult for the GAN-GAN to learn to model DCGAN with very small amounts of data. This is expected, as small perturbations to the weights in the first layers of the significantly deeper DCGAN model can significantly alter the output of the final layer. While it is likely that our approach would be viable with a larger dataset of DCGAN snapshots, this work was conducted with limited personal hardware. As such, we opted to use a small fully connected model and were unable to test GAN-GANs over DCGAN variants in detail.

4. Conclusion

We train a GAN over training snapshots of small, fully connected MNIST GANs and demonstrate that generative models are capable of learning useful representations of other generative models. In short: GAN you do the GAN GAN? Yes you GAN!

References

Deutsch, L. Generating Neural Networks with Neural Networks. *arXiv e-prints*, art. arXiv:1801.01952, Jan 2018.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1406.2661, Jun 2014.

Ha, D., Dai, A., and Le, Q. V. HyperNetworks. *arXiv e-prints*, art. arXiv:1609.09106, Sep 2016.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, art. arXiv:1412.6980, Dec 2014.

Liu, J., Tripathi, S., Kurup, U., and Shah, M. Make (Nearly) Every Neural Network Better: Generating Neural Network Ensembles by Weight Parameter Resampling. *arXiv e-prints*, art. arXiv:1807.00847, Jul 2018.

Radford, A., Metz, L., and Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1511.06434, Nov 2015.

Suarez, J. Language modeling with recurrent highway hypernetworks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 3267–3276. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6919-language-modeling-with-recurrent-highway-hypernet pdf.

Xu, B., Wang, N., Chen, T., and Li, M. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv e-prints*, art. arXiv:1505.00853, May 2015.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv e-prints*, art. arXiv:1703.10593, Mar 2017.