HyperGAN: Supplementary Material

A. Appendix

A.1. Generated Filter Examples

We show the first filter in 25 different networks generated by the HyperGAN to illustrate their difference in Fig. 1. It can be seen that qualitatively HyperGAN learns to generate classifiers with a variety of filters.

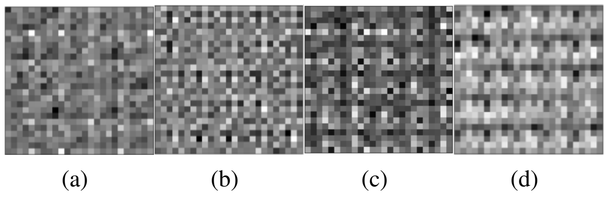

Figure 1. Convolutional filters from MNIST classifiers sampled from HyperGAN. For each image we sample the same 5x5 filter from 25 separate generated networks. From left to right: figures a and b show the first samples of the first two generated filters for layer 1 respectively. Figures c and d show samples of filters 1 and 2 for layer 2. We can see that qualitatively, HyperGAN learns to generate classifiers with a variety of filters.

A.2. Outlier Examples

In Figure 2 we show images of examples which do not behave like most of their respective distribution. On top are MNIST images which HyperGAN networks predict to have high entropy. We can see that they are generally ambiguous and do not fit with the rest of the training data. The bottom row shows notMNIST examples which score with low entropy according to HyperGAN. It can be seen that these examples look like they could come from the MNIST training distribution, making HyperGAN's predictions reasonable

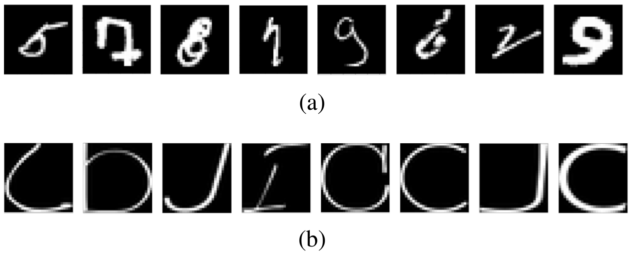

Figure 2. Top: MNIST examples to which HyperGAN assigns high entropy (outlier). Bottom: Not-MNIST examples which are predicted with low entropy (inlier)

A.3. HyperGAN Network Details

In tables 1 and 2 we show how the latent points are transformed through the generators to become a full layer of parameters. For a MNIST based HyperGAN we generate layers from small latent points of dimensionality 128. For CIFAR-10 based HyperGANs we use a larger dimensionality of 256 for the latent points.

Table 1. MNIST HyperGAN Target Size

Layer	Latent size	Output Layer Size
Conv 1	128 x 1	32 x 1 x 5 x 5
Conv 2	128 x 1	32 x 32 x 5 x 5
Linear	128 x 1	512 x 10

Table 2. CIFAR-10 HyperGAN Target Size

Layer	Latent Size	Output Layer Size
Conv 1	256 x 1	16 x 3 x 3 x 3
Conv 2	256 x 1	32 x 16 x 3 x 3
Conv 3	256 x 1	32 x 64 x 3 x 3
Linear 1	256 x 1	256 x 128
Linear 2	256 x 1	128 x 10

A.4. Diversity with Neither Mixer nor Discriminator

We run experiments on both MNIST and CIFAR-10 where we remove both the mixer and the discriminator. Tables 4 and 3 show statistics of the networks generated by HyperGAN using only independent Gaussian samples to the generators. In this setting, HyperGAN learns to generate only a very small distribution of parameters.

HyperGAN w/o (Q, D) - CIFAR-10					
	Conv1	Conv2	Conv3	Linear1	Linear2
Mean	1.87	16.83	9.35	10.66	20.35
σ	0.11	2.44	1.02	0.16	0.76
Standard Training - CIFAR-10					
	Conv1	Conv2	Conv3	Linear1	Linear2
Mean	5.13	15.19	16.15	11.79	2.45
σ	1.19	4.40	4.28	2.80	0.13

Table 3. Statistics on the layers of networks sampled from HyperGAN without the mixing network or discriminator, compared to 10 standard networks trained from different random initializations

	HyperGAN w/o (Q, D) - MNIST			Standard Training - MNIST		
	Conv1	Conv2	Linear	Conv1	Conv2	Linear
Mean	10.79	106.39	14.81	27.05	160.51	5.97
σ	0.58	0.90	0.79	0.31	0.51	0.06

Table 4. Statistics on the layers of a population of networks sampled from HyperGAN, compared to 10 standard networks trained from different random initializations. Without the mixing network or the discriminator, HyperGAN suffers from a lack of diversity

A.5. HyperGAN Diversity on Adversarial Examples

As an ablation study, in Fig. 3 we show the diversity of the HyperGAN predictions against adversarial examples generated to fool one network. It is shown that while those examples can fool $50\% - 70\%$ of the networks generated by HyperGAN, they usually never fool all of them.

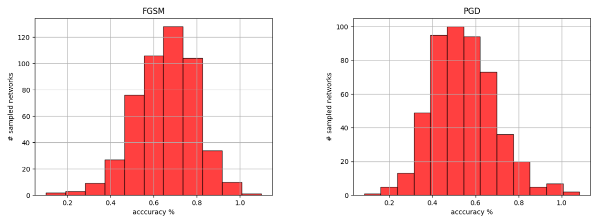

Figure 3. Diversity of predictions on adversarial examples. FGSM and PGD examples are created against a network generated by HyperGAN, and tested on 500 more generated networks. FGSM transfers better than PGD, though both attacks fail to cover the distribution learned by HyperGAN

A.6. Black Box Adversarial Examples

In addition to the white box attacks performed in section 4, we show here the results of HyperGAN on black box attacks. In the black box setting that we consider, the attack only has access to the $\arg\max$ of the softmax probabilities. We test HyperGAN against a powerful decision attack called the Boundary Attack. The Boundary Attack begins from a large adversarial perturbation, then reduces the magnitude of the perturbation while retaining an adversarial prediction. We use the foolbox toolbox as before, with a step size of 0.01 and a step multiplier of 1.5. We report defense robustness in terms of number of calls (model evaluations) required by the attack to fool the classifier. We measure at [100, 500, 1000, 2000, 5000] calls, and see that HyperGAN performs similarly well as on white box attacks.

It is important to note that the size of the perturbation decreases with more calls. We therefore expect that robustness decreases as calls increase.

Table 5. HyperGAN performance on Black box attacks

Ensemble Size	100	500	1000	2000	5000
5 nets	1.85	1.79	1.75	1.73	1.55
10 nets	1.88	1.79	1.76	1.76	1.63
100 nets	1.92	1.80	1.75	1.76	1.70
1000 nets	1.98	1.81	1.78	1.75	1.72

A.7. Effectiveness of Prior Matching

In our work we encourage diversity by regularizing the mixer posterior $Q(z|s)$ to be well-distributed according to a high entropy distribution (a isotropic Gaussian in practice). We do so by sampling points from $Q(z|s)$ and estimate their distance from a prior P with a discriminator $\mathcal{D}$. Regularizing the intermediate representation differs from standard variational inference, where samples from the posterior (generator outputs) are regularized to stay close to the prior. As stated in section 1, by regularizing the input samples, we retain flexibility in our generators to learn a more complex distribution over parameters than a unimodal high-dimensional Gaussian. To validate the effectiveness of the regularization we perform a normality test on 1000 samples from $Q(z|s)$, drawn for each layer for a 3 layer target model.

Table 6. Normality test on points from $Q(z|s)$

Layer	Mean	Std
Conv 1	0.085	1.01
Conv 2	0.014	1.20
Linear	0.091	0.73

We show the results from the normality test in table 6. We can see that the induced distribution is approximately normal with means close to 0. Furthermore, the mixer outputs distributions with different variance for each layer. The distribution with the highest variance corresponding to the second convolutional layer is what we expect, given that the representations of the final convolutional layer in a CNN are the most diverse.