MIXED MEMBERSHIP GENERATIVE ADVERSARIAL NETWORKS

Yasin Yazici[⋆] *Bruno Lecouat*[⋆] *Kim Hui Yap*[†] *Stefan Winkler*[‡]
Georgios Piliouras[§] *Vijay Chandrasekhar*[⋆] *Chuan-Sheng Foo*[⋆]

[⋆] Institute for Infocomm Research (I2R)
[†] Nanyang Technological University (NTU)
[‡] National University of Singapore (NUS)
[§] Singapore University of Technology and Design (SUTD)

ABSTRACT

GANs are designed to learn a single distribution, though multiple distributions can be modeled by treating them separately. However, this naive implementation does not consider overlapping distributions. We propose Mixed Membership Generative Adversarial Networks (MMGAN) analogous to mixed-membership models that model multiple distributions and discover their commonalities and particularities. Each data distribution is modeled as a mixture over a common set of generator distributions, and mixture weights are automatically learned from the data. Mixture weights can give insight into common and unique features of each data distribution. We evaluate our proposed MMGAN and show its effectiveness on MNIST and Fashion-MNIST with various settings.

Index Terms— generative models, generative adversarial networks, mixture models, mixture membership models

1. INTRODUCTION

Generative Adversarial Networks (GANs) [1] learn a function that can be used to draw samples efficiently from the learnt probability distribution. Informally, it achieves this by pitting a discriminator and a generator against each other as an adversarial game. Due to enormous interest, GANs have been improved substantially over the past few years [2–5].

GANs are designed to learn a single distribution, though multiple distributions can be modeled by treating them separately. However, this naive implementation does not consider relationships between distributions. How then can we use GANs to model multiple distributions in a way that discovers their common and unique aspects? Fig. 1 shows an example of two data distribution, which we would like to model by considering their commonalities and particularities. Indeed, this setting corresponds to well-known mixed membership models, where individual samples may belong to more than one group [6–8].

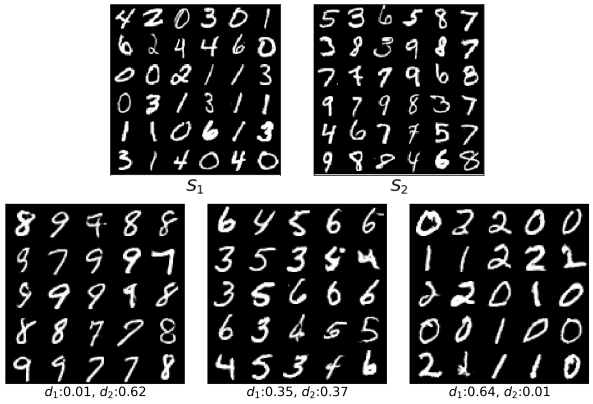

Fig. 1. (Top) samples from two data distributions (S1: digits 0-6, S2: digits 3-9). (Bottom) MMGAN modeling: the middle component is the common part (digits 3-6), while the left and right ones are unique aspects: digits 7-9 and digits 0-2.

Unlike most previous mixed membership models, our method does not explicitly optimize a likelihood function, but is an implicit generative model based on GANs. This provides two major advantages: (i) the objective function can be optimized without approximation such as variational approaches [6], and (ii) there is no need to use conjugate priors to the likelihood, increasing flexibility.

We propose Mixed Membership GANs (MMGAN) which model multiple distributions and discover their commonalities and particularities. MMGAN specifically targets high-dimensional image distributions and is one of the few instances of a mixed-membership model on continuous data. Each data distribution is modeled as a mixture over a common set of generator distributions; mixture weights are automatically learned from the data. Depending on the mixture weights, the shared components capture the common as-

Y. Yazici is the contact author. B. Lecouat and V. Chandrasekhar were part of I2R during the project. S. Winkler is now also with ASUS AICS. This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funds (Grant No. A20H6b0151). The computational work for this article was performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

pects of the distributions, while non-shared ones capture their unique aspects. We evaluate our method on MNIST and Fashion MNIST with compelling results.

2. RELATED WORK

Multi-generator/discriminator GAN. Multiple generators/ discriminators have been used in order to solve various issues with GAN. [9, 10] modeled a single distribution with multiple generators to capture different modes of the distribution. [11–13] utilized multiple discriminators to address mode collapse and optimization stability.

Mixture Distribution with GANs. Some of the earlier works considered multiple generators as mixture of distributions to model a single distribution [9,10]. [14] used mixture of GANs to discover clusters in a distribution.

Mixed Membership Models. These models are useful where a sample does not belong to a single cluster but multiple ones. It has been used for topic analysis [6], genomics discovery [15], scene clustering [8] etc. Their applications on image datasets are limited. Recently, [16, 17] have utilized adversarial learning in their model for topic analysis.

Comparison. We use multiple generators/discriminators, however the motivation of our paper is different, namely to learn mixed-membership modeling. Our approach is also different from mixture distribution with GANs, as they do not aim to learn our aforementioned objective. Conceptually, our modeling is similar to class conditional mixture of distributions with shared components [18,19] and mixed-membership models; though our modeling is with GANs, which makes it possible to model high-dimensional data distributions like images. [20] is perhaps the most similar to ours. However, their motivation, method and experiments are different from ours. Similarly, [16, 17] have utilized adversarial objective for topic models, however their objective, architecture, data modality, and task are different than ours. Specifically, their methods are tuned for NLP, on discrete text data, while we present a model for continuous image distributions.

3. METHOD

Background. GANs are an adversarial game between a discriminator (D) and a generator (G):

$$\min_G \max_D V(D, G) \tag{1}$$

$$V(D, G) = \mathbb{E}_{p_d(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{p_g(\boldsymbol{x})}[\log(1 - D(\boldsymbol{x}))] \tag{2}$$

where p_d is the data distribution, and p_g is the generator distribution. The discriminator assesses a pseudo-divergence between $p_d(\boldsymbol{x})$ and $p_g(\boldsymbol{x})$. The discriminator maximizes the divergence, while the generator minimizes it. In this way, the generator learns to mimic the data distribution implicitly. [1] shows that, under certain assumptions, for a fixed optimal D, minimizing Eq. 2 for G would lead to $p_g(\boldsymbol{x}) = p_d(\boldsymbol{x})$.

Multi-distribution GAN. The value function, Eq. 1, can be generalized for n distributions as follows:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D}, \mathcal{G}) \tag{3}$$

$$\begin{aligned} V(\mathcal{D}, \mathcal{G}) = &\frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{p_{d_i}}[\log D_i(\boldsymbol{x})] + \\ &\frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{p_{g_i}}[\log(1 - D_i(\boldsymbol{x}))] \end{aligned} \tag{4}$$

where $\mathcal{D} = \{D_1, ..., D_n\}$, $\mathcal{G} = \{G_1, ..., G_n\}$, p_{d_i} is the i-th data distribution and p_{g_i} is the i-th generator distribution. Note that D_i and G_j in the above equation interact with one another only when $i = j$. This makes learning one distribution independent from the others. Due to this property, at equilibrium $p_{d_i}(\boldsymbol{x}) = p_{g_i}(\boldsymbol{x})$ by *Proposition 1* from [1].

Mixture Membership GAN (Proposed Approach). The above objective does not consider possible similarities between data distributions. To account for this we model each data distribution as a mixture over a collection of generators (components) that are shared across multiple data distributions. We effectively replace p_{g_i} as follows:

$$p_{g_i} = \sum_{k=1}^{K} \alpha_k^{(i)} p_{\tilde{g}_k} \quad \text{with} \quad \sum_{k=1}^{K} \alpha_k^{(i)} = 1 \tag{5}$$

where $\alpha_k^{(i)}$ is a mixture weight for the k-th component (with distribution $p_{\tilde{g}_k}$) and i-th distribution. In this way, each data distribution is modeled as a mixture of distributions. As $p_{\tilde{g}_i}$ are shared for all data distributions, some of them cover common parts, while others cover unique parts depending on $\alpha_k^{(i)}$. Each $p_{\tilde{g}_k}$ learns only a sub-population of distributions, and their combination at different amounts create data distributions. By substituting Eq. 5 into the second term of Eq. 4:

$$\mathbb{E}_{p_{g_i}} \log(1 - D_i(\boldsymbol{x})) = \sum_{k=1}^{K} \alpha_k^{(i)} \mathbb{E}_{p_{\tilde{g}_k}} \log(1 - D_i(\boldsymbol{x})) \tag{6}$$

Then, by substituting Eq. 6 into the second term of Eq. 4, we come up with the MMGAN objective:

$$\min_{\mathcal{G}, \mathcal{W}} \max_{\mathcal{D}} V(\mathcal{D}, \mathcal{G}, \mathcal{W}) \tag{7}$$

$$\begin{aligned} V(\mathcal{D}, \mathcal{G}, \mathcal{W}) = &\frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{p_{d_i}(\boldsymbol{x})}[\log D_i(\boldsymbol{x})] + \\ &\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \alpha_k^{(i)} \mathbb{E}_{p_{\tilde{g}_k}(\boldsymbol{x})}[\log(1 - D_i(\boldsymbol{x}))] \end{aligned} \tag{8}$$

$$\alpha_k^{(i)} = \frac{\exp(\omega_k^{(i)})}{\sum_{j=1}^{K} \exp(\omega_j^{(i)})} \tag{9}$$

where $\mathcal{D} = \{D_1, ..., D_n\}$, $\mathcal{G} = \{G_1, ..., G_K\}$, $\mathcal{W} = \{\boldsymbol{\omega}^{(1)}, ..., \boldsymbol{\omega}^{(n)}\}$, $\boldsymbol{\omega}^{(i)} \in \mathbb{R}^K$ are parameters for mixture weights for the i-th distribution, p_{d_i} is the i-th data distribution, $p_{\tilde{g}_k}$ is the k-th generator (component) distribution.

Effect of $\alpha_k^{(i)}$. As $\boldsymbol{\alpha}^{(i)}$ is a probability simplex, it affects the relative scale of the loss for each component. Also, as $\boldsymbol{\omega}^{(i)}$ is a minimizer of Eq. 7, $\alpha_k^{(i)}$ would be minimized more than the other entries if the k-th component yields a higher loss than the other components for the i-th distribution. This can be seen better from the gradients; as $\frac{\partial V(\mathcal{D},\mathcal{G},\mathcal{W})}{\alpha_k^{(i)}} \propto \mathbb{E}_{\boldsymbol{x}\sim p_{\tilde{g}_k}}[\log(1 - D_i(\boldsymbol{x}))]$, the error of $\alpha_k^{(i)}$ is proportional to loss of k-th generator over the i-th discriminator. In other words, if the k-th generator yields a high loss from the i-th discriminator, then contribution of the k-th generator would be scaled down.

Enforcing disjointness. We wish to make each component ($p_{\tilde{g}_k}$) disjoint, so that they capture different sub-populations of the overall distributions. To encourage disjointness, we include a classifier into the objective that aims to separate each component from the others:

$$\max_{C,\mathcal{G}} \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log C(y_k|G_k(\boldsymbol{z};\theta_k);\phi_c)] \quad (10)$$

where y_k is a categorical label of $\boldsymbol{x} \sim p_{\tilde{g}_k}(\boldsymbol{x})$, and C is a classifier that outputs the class distribution over the components. With this objective, the classifier tries to separate samples from different components, while the components maximize the difference among them, similar to [21]. The combined objective becomes:

$$\min_{C,\mathcal{G},\mathcal{W}} \max_{\mathcal{D}} V(\mathcal{D},\mathcal{G},\mathcal{W}) - \frac{\lambda}{K} \sum_{k=1}^{K} \mathbb{E}_{p_{\tilde{g}_k}}[\log C(y_k|\boldsymbol{x})] \quad (11)$$

where λ is a hyper-parameter balancing the two terms.

Sampling process. Once the MMGAN is trained, samples can be drawn from the generator distribution as follows:

- For each data distribution i:
  - Draw mixture assignment $k \sim Cat(\boldsymbol{\alpha}^{(i)})$
  - Draw prior distribution $\boldsymbol{z} \sim p_{\boldsymbol{z}}$
    - Draw data samples $\boldsymbol{x} \sim G_k(\boldsymbol{z})$

4. EXPERIMENTS

Datasets. We investigate the working dynamics of our method on MNIST and Fashion-MNIST [22]. With these datasets, we construct 2- and 3-distributions by using the class information from the datasets as per Table 1. We never use a sample twice for the splits to avoid trivial solutions.

Network Architecture & Training. Instead of independent generators, we use a single conditional generator with K conditions due to efficiency. Each one is modeled as $G_i = G(z, c = i; \theta)$, where c is a condition whose i-th index generates G_i, and θ are the network parameters. We interchangeably use generator and component to refer to a specific G_i. There are n discriminators in an n-distribution game. They share all but the last layer with the classifier; the classifier includes one more layer on top of this with K-dimensional output. We use DCGAN architecture [2]. Exponential moving average is used over generator parameters as in [4, 23]. Conditioning of G is similar to [24]. We set $K = 2^n - 1$ in our experiments; note that the model may collapse some components during training by assigning zero weight to them. We use the ADAM [25] optimizer for all networks with $\alpha = 0.0002$, $\beta_1 = 0.0$ and $\beta_2 = 0.9$. For mixture weights $\alpha = 0.0004$. The optimization of discriminator and generator follows the alternating update rule with update ratio 1:1. We run the experiments for 100k iterations. For each component the batch size is 32, and $\lambda = 0.5$.

Quantitative Results. As MMGAN is an unsupervised learning model, its quantitative evaluation is not trivial. To support our qualitative experiments, we quantify the rate of correct generation (accuracy) for each component, i.e. if a specific item appears in the correct component (generator). As the correspondence between generator components and their true labels is not known beforehand, we fix mixture weights with true priors. To measure accuracy, we use pre-trained classifiers for MNIST and Fashion-MNIST. As there is no comparable method in the literature, we compare against a random baseline: the probability of a random sample appearing in the correct component.

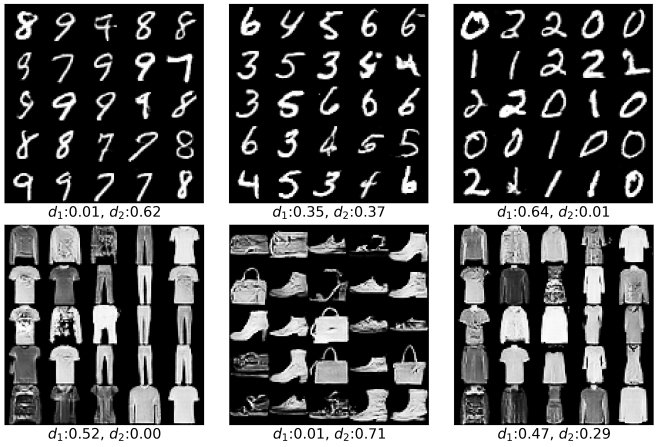

Fig. 2. MNIST and Fashion-MNIST results for case A.

5. RESULTS

Qualitative Results. Figs. 2, 3, and 4 show the results with learned mixture weights for cases A, B, and C respectively (cf. Table 1). We draw a number of samples from each generator to show what they synthesize. The weight d_i in each subfigure indicates the value of the mixture weight, $\alpha_k^{(i)}$ in Eq. 8 for a component. The Figures demonstrate that each generator captures a part of a distribution that corresponds to unique or common parts of the distributions. For example, MNIST of Fig. 2 illustrates 3 components where the first and

Table 1. (Top) Configurations of the datasets. (Bottom) Correspondence of labels for Fashion-MNIST.

Case	Distributions	Sets
A	2	$S_1 = \{0, 1, 2, 3, 4, 5, 6\}$, $S_2 = \{3, 4, 5, 6, 7, 8, 9\}$
B	3	$S_1 = \{0, 3, 5, 6\}$, $S_2 = \{1, 4, 5, 6\}$, $S_3 = \{2, 3, 4, 6\}$
C	3	$S_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $S_2 = \{3, 4, 5, 6, 7, 8, 9\}$, $S_3 = \{6, 7, 8, 9\}$

Label	0	1	2	3	4	5	6	7	8	9
Item	T-shirt/top	Trousers	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot

Table 2. Accuracy of each component for MNIST and Fashion-MNIST for cases A, B, and C vs. random baselines. Set operations for case A: $r_1 = S_1 \setminus S_2$, $r_2 = S_2 \setminus S_1$, $r_3 = S_1 \cap S_2$. Set operations for cases B and C: $r_1 = S_1 \setminus (S_2 \cup S_3)$, $r_2 = S_2 \setminus (S_1 \cup S_3)$, $r_3 = S_3 \setminus (S_1 \cup S_2)$, $r_4 = (S_1 \cap S_3) \setminus S_2$, $r_5 = (S_2 \cap S_3) \setminus S_1$, $r_6 = (S_1 \cap S_2) \setminus S_3$, and $r_7 = S_1 \cap S_2 \cap S_3$.

Dataset	Case	r_1	r_2	r_3	r_4	r_5	r_6	r_7	Avg	Random
MNIST	A	99.69	98.86	83.40	n/a	n/a	n/a	n/a	93.98	0.5
F-MNIST	A	90.15	86.48	80.92	n/a	n/a	n/a	n/a	85.85	0.5
MNIST	B	99.32	100.0	96.05	98.75	98.14	99.56	99.36	98.74	0.25
F-MNIST	B	71.86	98.07	68.45	71.04	93.17	91.54	18.02	73.16	0.25
MNIST	C	99.12	n/a	n/a	n/a	n/a	93.08	93.64	95.28	0.61
F-MNIST	C	94.41	n/a	n/a	n/a	n/a	83.17	67.5	81.69	0.61

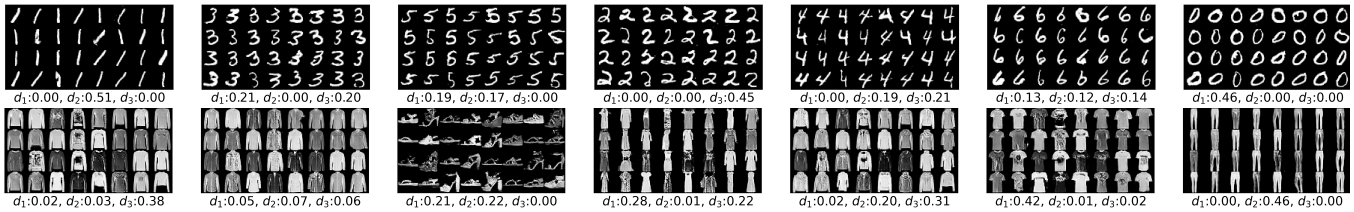

Fig. 3. MNIST and Fashion-MNIST results for case B.

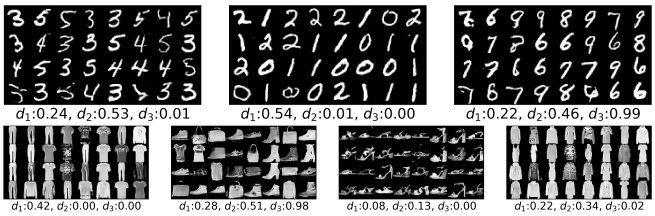

Fig. 4. MNIST and Fashion-MNIST results for case C.

third components exclusively contribute to unique parts of the distribution, while the second component is the common part of both distributions that is digits of $3, 4, 5, 6$. As our method is unsupervised, it may not learn the smallest possible number of components. For example, in Fig. 4, the minimum number of components is 3, but Fashion-MNIST results converged to 4 components where the last two could have been combined.

Quantitative Results. Table 2 lists quantitative results for the experiments above, where mixture weights are initialized with priors as mentioned before. For ease of interpretation, we have labeled components with set operations. For example, for case A of MNIST, $r_1 = S_1 \setminus S_2 = \{0, 1, 2\}$ and the component generates digits $\{0, 1, 2\}$ correctly 99.69% of the time. As expected, the average score for each setting is highly correlated with the qualitative interpretations above. All the scores are also significantly above the random baseline, which demonstrates the effectiveness of our model. Other than the average scores, we can analyze how well individual components are modeled and how they compare to one another. Intuitively, simpler components are modeled better, e.g. case B of MNIST $S_2 \setminus (S_1 \cup S_3)$ is digit 1 which obtains a perfect score, or case B of Fashion-MNIST $S_2 \setminus (S_1 \cup S_3)$ is *Trousers* which outperforms the other items.

6. CONCLUSIONS

We have proposed a novel mixed-membership model based on GANs that can discover particularities and commonalities over multiple data distributions. MMGAN models each data distribution with a mixture of common generator distributions. As the generators are shared across multiple true data distributions, highly shared generators (reflected through mixture weights) capture common aspects of the distributions, while non-shared ones capture unique aspects. The shared generators and mixture weights are learned end-to-end together with other parameters. We have successfully trained MMGAN on MNIST and Fashion-MNIST datasets to show its effectiveness.

We hope to scale MMGAN to higher resolution large number of distributions in the future, which will open up new directions for analyzing image collections, analogous to what topic models have enabled for document collections.

7. REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, 2014.

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015.

[3] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *Proc. International Conference on Learning Representations*, 2018.

[4] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *International Conference on Learning Representations*, 2018.

[5] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Machine Learning Research*, vol. 3, 2003.

[7] E. Erosheva, "Bayesian estimation of the grade of membership model," in *Bayesian Statistics 7*, 2003.

[8] L. Fei-Fei and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[9] Q. Hoang, T. D. Nguyen, T. Le, and D. Q. Phung, "Multi-generator generative adversarial nets," *CoRR*, vol. abs/1708.02556, 2017.

[10] A. Ghosh, V. Kulharia, V. P. Namboodiri, P. H. S. Torr, and P. K. Dokania, "Multi-agent diverse generative adversarial networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[11] I. P. Durugkar, I. Gemp, and S. Mahadevan, "Generative multi-adversarial networks," *CoRR*, vol. abs/1611.01673, 2016.

[12] B. Neyshabur, S. Bhojanapalli, and A. Chakrabarti, "Stabilizing GAN training with multiple random projections," *CoRR*, vol. abs/1705.07831, 2017.

[13] F. Juefei-Xu, V. N. Boddeti, and M. Savvides, "Gang of GANs: Generative adversarial networks with maximum margin ranking," *CoRR*, vol. abs/1704.04865, 2017.

[14] Y. Yu and W.-J. Zhou, "Mixture of gans for clustering," in *Proc. 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[15] J. K. Pritchard, M. Stephens, and P. Donnelly, "Inference of population structure using multilocus genotype data," *Genetics*, vol. 155, no. 2, 2000. [Online]. Available: https://www.genetics.org/content/155/2/945

[16] T. Masada and A. Takasu, "Adversarial learning for topic models," in *ADMA*, 2018.

[17] R. Wang, X. Hu, D. Zhou, Y. He, Y. Xiong, C. Ye, and H. Xu, "Neural topic modeling with bidirectional adversarial training," in *Proc. 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[18] M. K. Titsias and A. C. Likas, "Shared kernel models for class conditional density estimation," *IEEE Trans. Neural Networks*, vol. 12, no. 5, 2001.

[19] M. K. Titsias and A. Likas, "Class conditional density estimation using mixtures with constrained component sharing," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, 2003.

[20] T. Kaneko, Y. Ushiku, and T. Harada, "Class-distinct and class-mutual image generation with gans," *CoRR*, vol. abs/1811.11163, 2018.

[21] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung, "MGAN: Training generative adversarial nets with multiple generators," in *Proc. International Conference on Learning Representations*, 2018.

[22] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017.

[23] Y. Yazıcı, C.-S. Foo, S. Winkler, K.-H. Yap, G. Piliouras, and V. Chandrasekhar, "The unusual effectiveness of averaging in GAN training," in *Proc. International Conference on Learning Representations*, 2019.

[24] T. Miyato and M. Koyama, "cGANs with projection discriminator," in *Proc. International Conference on Learning Representations*, 2018.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.