TRAINING GENERATIVE ADVERSARIAL NETWORKS WITH WEIGHTS

Yannis Pantazis[1], *Dipjyoti Paul*[2], *Michail Fasoulakis*[3], *Yannis Stylianou*[2]

[1]Institute of Applied and Computational Mathematics, FORTH, Greece
[2]Department of Computer Science, University of Crete, Greece
[3]Institute of Computer Science, FORTH, Greece

ABSTRACT

The impressive success of Generative Adversarial Networks (GANs) is often overshadowed by the difficulties in their training. Despite the continuous efforts and improvements, there are still open issues regarding their convergence properties. In this paper, we propose a simple training variation where suitable weights are defined and assist the training of the Generator. We provide theoretical arguments which indicate that the proposed algorithm is better than the baseline algorithm in the sense of creating a stronger Generator at each iteration. Performance results showed that the new algorithm is more accurate and converges faster in both synthetic and image datasets resulting in improvements ranging between 5% and 50%.

Index Terms— Generative adversarial networks, multiplicative weight update method, training algorithm.

1. INTRODUCTION

A fully data-driven paradigm in conducting science has been emerged during the last years with the advent of GANs [1]. A GAN offers a new methodology for drawing samples from an unknown distribution where only samples from this distribution are available making them one of the hottest areas in machine learning/artificial intelligence research. Indicatively, GANs have been successfully utilized in (conditional) image creation [2, 3, 4], generating very realistic samples [5, 6], speech signal processing [7, 8], natural language processing [9] and astronomy [10], to name a few.

A GAN is a two-player *zero-sum game* [1, 11] between a Discriminator and a Generator, both being powerful neural networks. They are simultaneously trained to achieve a *Nash equilibrium*, where the Discriminator cannot distinguish the real and the fake samples while the Generator has learned the unknown distribution. It is well-known that the training procedure of GANs often fails and several specific heuristics and hacks have been devised [12] along with general-purpose acceleration techniques such as batch normalization [13]. To alleviate the difficulties of training, extensions and generalizations stemming from the utilization of a different loss function has been proposed. For instance, f-GAN [14] is a generalization where the f-divergence is used instead of the *Shannon-Jensen divergence* of the original GAN. Another widely-applied extension is *Wasserstein GAN* [15] which has been further improved in [16]. On the other hand, there are relatively few studies that aim directly to improve the convergence speed of training of an existing GAN.

In this paper, instead of proposing a new GAN architecture or a new GAN loss function we propose a new training algorithm inspired by the *multiplicative weight update method* (MWUM) [17]. Our goal is to improve the training of the Generator by transferring ideas from Game Theory. Intuitively, the new algorithm puts more weight to fake samples that are more probable to fool the Discriminator and simultaneously reduces the weight of samples that are confidently discriminated as fake. Our contributions are summarized as follow: (i) By adding weights to the training of GANs, we manage to improve the training performance with minor additional computational cost. The new approach is called *Weighted GAN* (WeGAN). (ii) We provide rigorous arguments that the weights of WeGAN locally reduces the loss function more or at least as much as the equally-weighted stochastic gradient descent for the Generator. (iii) The proposed algorithm is not specific to vanilla GAN [1], but it is directly transferable to other extensions such as *conditional GANs*, *Wasserstein GAN* and *f-GAN*. This is an important generalization property of WeGAN.

Before proceeding, it is worth-noting that training methods utilizing weights for the Generator have been recently proposed [18, 9, 19]. These methods are essentially equivalent to each other since they assign importance weights to the generated samples in order to obtain a tighter lower bound for their variational formula. However, importance weights GAN (IWGAN) cannot be applied to any type of objective function and additionally they might diverge due to their unboundedness. We implemented IWGAN and present its performance in the Results section comparing it to our algorithm.

2. PRELIMINARIES

GAN formulation. Let G be the Generator and D be the

Emails: pantazis@iacm.forth.gr, {dipjyotipaul, yannis}@csd.uoc.gr, mfasoul@ics.forth.gr

978-9-0827-9703-9/19/$31.00 ©2019 IEEE

Discriminator of the GAN [1]. $D(x)$ computes the probability of sample x being real while $G(z)$ is the sample produced by the Generator given noise input z. In order to be trained, the following objective function of the two-player *zero-sum game* has to be optimized:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))],$$

where p_{data} is the distribution to be learned, while p_z is the noise distribution. Typically, an optimum (*Nash equilibrium*) of this *zero-sum game*, which is a saddle point, is estimated using *stochastic gradient descent*. As it was proved in [1], the global optimum of this *zero-sum game* is the point where $D(x) = 1/2$ for any sample x and the Generator generates samples according to the real distribution.

MWUM basics. MWUM is a classic algorithmic technique with numerous applications. The main idea behind this method is the existence of a number of "experts" that give some kind of advice to a decision maker. To any "expert" a specific weight is assigned and the initial weights are equal for any "expert". Then, the decision maker takes the decision according to the advice of the "experts" taking into account the weight of any of them. After this the weights are multiplicatively updated according to the performance of the advice of any individual "expert"; increasing the weights of the "experts" with good performance and decreasing them otherwise and so on. We continue with the description of our algorithm and the connection to this method.

3. WEIGHTED GAN ALGORITHM

The proposed algorithm presented in Fig. 1 is a modification of the original GAN training algorithm. Inspired by the MWUM, instead of equally-weighted 'fake' samples, we assign a weight to each sample (the "expert" in MWUM) which multiplies the respective gradient term of the Generator. The weighting aims to put more strength to samples that fool the Discriminator and thus are closer to the real data. Indeed, when $D(G(z)) = 0$ and the Discriminator understands that the sample is fake the weight decreases by a factor $\eta \in (0, 1]$. On the other hand, when $D(G(z)) = 1$ the weight remains the same and after the normalization step it has a value greater or equal than the previous one. Notice also that the weights in Algorithm 1 depend only of the current value of the Discriminator while in the standard MWUM the weights are updated cumulatively. This modification was necessary because the input samples are different at each iteration. Indeed, new samples are generated and there is no obvious map between the current samples and the samples from the previous iteration.

3.1. Theoretical properties of WeGAN algorithm

A key assumption of our algorithm as well as in other weighting algorithms is that the Discriminator is faithful in the sense that it produces sound decisions for both real and fake samples. Quantitatively, it means that the Discriminator should return on average values above 0.5 when the sample comes from the real distribution and below 0.5 when fake samples are fed to the Discriminator. Next, we show that for a fixed Discriminator, the optimal Generator with weights as in Algorithm 1 achieves lower or equal loss value than the optimal Generator with equally-weighted samples. Hence, we expect that the inferred Generator is stronger favorably affecting the convergence properties.

Algorithm 1

for *number of iterations* **do**
  for *k steps* **do**
    Sample $\{x_1, \ldots, x_m\}$ from the data distribution $p_{data}(x)$.
    Sample $\{z_1, \ldots, z_m\}$ from the input distribution $p_z(z)$.
    Update the Discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x_i\right) + \log\left(1 - D\left(G\left(z_i\right)\right)\right)\right].$$

  end
  Sample $\{z_1, \ldots, z_m\}$ from the input distribution $p_z(z)$.
  Compute the unnormalized weights:

$$w_i = \eta^{(1-D(G(z_i)))},\ i = 1, ..., m.$$

  Normalize:

$$w_i = \frac{w_i}{\sum_{j=1}^{m} w_j},\ i = 1, ..., m.$$

  Update the Generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \sum_{i=1}^{m} w_i \log\left(1 - D\left(G\left(z_i\right)\right)\right).$$

end

Fig. 1. Stochastic gradient ascend/descent training of WeGAN. For a direct comparison with the original GAN, we follow the formulation of [1].

Theorem 1. *Fix Discriminator D and let $G^*_{D;w}$ and $G^*_{D;\frac{1}{m}}$ be the respective optimum Generator under weighted and equally-weighted loss function defined by*

$$L(G, D; w) = \frac{1}{m}\sum_{i=1}^{m} \log(D(x_i)) + \sum_{i=1}^{m} w_i \log(1 - D(G(z_i))).$$

Let the weight vector, w, be defined according to Algorithm 1 then

$$L(G^*_{D;w}, D; w) \leq L(G^*_{D;\frac{1}{m}}, D; \frac{1}{m}). \quad (1)$$

Proof. By definition, it holds for the optimum Generator that

$$L(G^*_{D;w}, D; w) \leq L(G^*_{D;\frac{1}{m}}, D; w).$$

If we prove that for any G, it holds that $L(G, D; w) \leq L(G, D; \frac{1}{m})$ when w is defined as in Algorithm 1 we are done because we get the desired result for $G = G^*_{D;\frac{1}{m}}$. Without loss of generality, we prove the case with $m = 2$ samples. Using a more elaborate but similar argument we can prove it for the general case.

Assuming that $D(G(z_1)) > D(G(z_2))$, it is easy to show that $w_1 > w_2$ and $\log(1 - D(G(z_1))) < \log(1 - D(G(z_2)))$. Next, let n, k be positive integers such that $w_1 = \frac{k}{2n} + \varepsilon_1$ and $w_2 = \frac{2n-k}{2n} + \varepsilon_2$, with ε_i be arbitrarily small constants for $i \in \{1, 2\}$. This is possible due to the fact that the set of rational numbers is a dense subset of real numbers. Since $w_1 > w_2$ implies $k > n$, then, it holds

$$
\begin{aligned}
& w_1 \log(1 - D(G(z_1))) + w_2 \log(1 - D(G(z_2))) + \varepsilon \\
=& \frac{1}{2n}\big[k \log(1 - D(G(z_1))) + (2n - k) \log(1 - D(G(z_2)))\big] + \varepsilon \\
=& \frac{1}{2n}\big[n \log(1 - D(G(z_1))) + n \log(1 - D(G(z_2))) \\
& + (k - n)(\log(1 - D(G(z_1))) - \log(1 - D(G(z_2))))\big] + \varepsilon \\
\leq& \frac{1}{2n}\big[n \log(1 - D(G(z_1))) + n \log(1 - D(G(z_2)))\big] + \varepsilon \\
=& \frac{1}{2} \log(1 - D(G(z_1))) + \frac{1}{2} \log(1 - D(G(z_2))) + \varepsilon,
\end{aligned}
$$

for arbitrarily small positive ε. Thus, we prove for $m = 2$ that

$$L(G, D; w) \leq L(G, D; \frac{1}{2}).$$

At equilibrium. It is straightforward to show that at the Nash equilibrium the weights of WeGAN are uniform. Indeed, it holds that $D(x) = 0.5$ for all x and thus

$$w_i = \frac{\eta^{1-D(G(z_i))}}{\sum_{j=1}^{m} \eta^{1-D(G(z_j))}} = \frac{\eta^{0.5}}{\sum_{j=1}^{m} \eta^{0.5}} = \frac{1}{m}\,.$$

This observation can serve either as a criterion to stop the training process or as an evaluation metric to assess whether or not the training process converged to an optimum. Monitoring the variance of the weights is the simplest statistic for both tasks.

WeGAN generalization. The proposed algorithm is not exclusive for vanilla GAN and it can be easily extended and applied to any variation of GANs that incorporates a Discriminator mechanism. Therefore, we do not propose just an extension of vanilla GAN but rather a novel training algorithm for general GANs. For instance, we could assign the same formula as in vanilla GAN for the weights for Wasserstein GAN. The presented theoretical analysis still holds for this case.

4. RESULTS

For a fair comparison, we evaluate the performance of the various training algorithms without changing the architecture of the networks. Moreover, with the exception of CIFAR, the presented results are averaged over 1000 iterations.

4.1. An illustrative example

We present a benchmark example where the new algorithm converges to the data distribution faster than vanilla GAN. The 'real' data are drawn from a mixture of 8 normal distributions with each of the 8 components being equally-probable. The mean values are equally-distributed on a circle with radius 3 and covariance matrix I_d. Moreover, both Generator and Discriminator are fully-connected neural networks with 2 hidden layers and 32 units per layer. The input random variable has a 2-dimensional standard normal while the output of the Discriminator is the sigmoid function.

The upper and middle plots of Fig. 2 show the relative improvement of WeGAN with respect of vanilla GAN for various values of η (circle, square & star lines) as a function of the number of epochs. The chosen performance metric is the maximum mean discrepancy (MMD) [20] which measures the closeness between the real data and the generated ones. The relative improvement is higher at the early stage when only $k = 1$ iteration in the training of the Discriminator is performed (upper plot of Fig. 2). In contrast, the highest relative improvement occurs closer to the convergence regime when $k = 5$ iterations in Discriminator's training are performed (middle plot). For comparison purposes we added IWGAN (dashed line) which also outperforms vanilla GAN but it is slightly worse than WeGAN with $\eta = 0.01$. Moreover, there were cases where IWGAN diverges because it produced a weight with infinite value. In the lower plot of Fig. 2, we present the relative performance improvement between the baseline training algorithm for the Wasserstein GAN and the respective weighted variation. We observe that improvements happen but they are less prominent. Additionally, higher values of η result in better performance which is the opposite situation when compared with the vanilla GAN.

4.2. MNIST

We extend our experiments on common benchmark MNIST image database of handwritten digits [21, 22]. In this experiment, a single hidden layer based fully connected neural network has been used for both Generator and Discriminator with 128 hidden units. Whereas, the input to Generator is set to 100 dimensional standard normal random variables. Two popular evaluation metrics i.e., Inception Score (IS) [12] and Fréchet Inception Distance (FID) [23] are used to quantitatively assess the performance of GANs. Both metrics assume access to a pre-trained classifier and provide an objective score based on the distribution of the sample that is to be evaluated. Overall relative performance, for IWGAN and various versions of WeGAN with respect to vanilla GAN in terms of IS (upper plot) and FID (lower plot) metrics, are presented in Fig. 3. Evidently, WeGAN algorithm outperforms standard vanilla GAN with relative improvement of almost 10% in IS and 30% in FID metrics. Results reveal that WeGAN with $\eta = 0.01$ has the best improvement when compared to other variations of η values which is consistent with

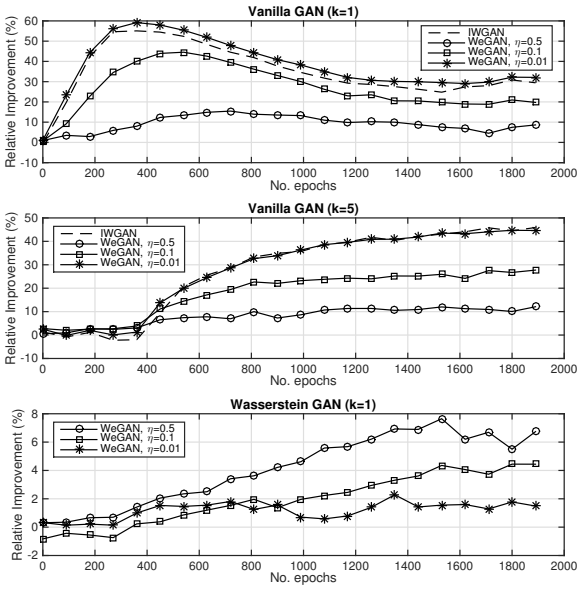

Fig. 2. Upper & Middle plot: Relative improvement as a function of the epochs in terms of mean MMD with respect to vanilla GAN for a mixture of 8 Gaussians. Lower values for η resulted in improved convergence of WeGAN (lines with circles, squares and stars). Lower plot: Similar to the other plots for Wasserstein GAN. Higher values for η gave faster convergence while IWGAN is not applicable.

the earlier reported results. By examining Fig. 3, we also observe that IWGAN achieves higher relative improvement in the early epochs, however, fails to maintain the performance as oppose to WeGAN at $\eta = 0.01$ which procures the best performance.

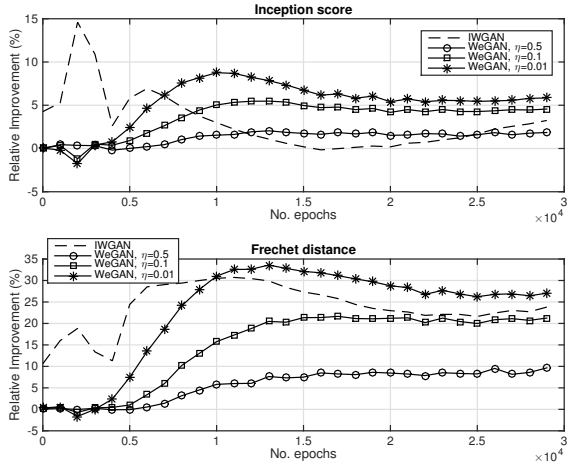

Fig. 3. Relative improvement as a function of the epochs in terms of IS (upper plot) and FID (lower plot) with respect to vanilla GAN for the MNIST digit dataset. As in the benchmark example, lower values for η result in improved convergence of the WeGAN.

4.3. CIFAR

CIFAR-10 is a well studied dataset of natural images [24]. We use this dataset to examine the performance of GANs. For the Generator, we use a deep convolutional network with a single linear layer followed by 3 convolutional layers. Whereas, the Discriminator has 4 convolutional layers and 1 linear layer at the end. Batch normalization is applied to both networks. The input noise with dimensionality of 100 is drawn from a uniform distribution. Fig. 4 shows IS (upper plot) and FID (lower plot) scores for the CIFAR-10 dataset in terms of relative improvement with reference to vanilla GAN. It can be observed that the proposed WeGAN with $\eta = 0.01$ is preferred over all respective weighted variations in IS score with 5–10% of improvement. Whereas, WeGAN with $\eta = 0.5$ & 0.1 both performs comparatively well in FID score. Unfortunately, the performance metrics produce conflicting outcomes making it hard to draw a clear conclusion for this dataset. We also evaluate IWGAN, however, its performance remains approximately the same against the baseline vanilla GAN.

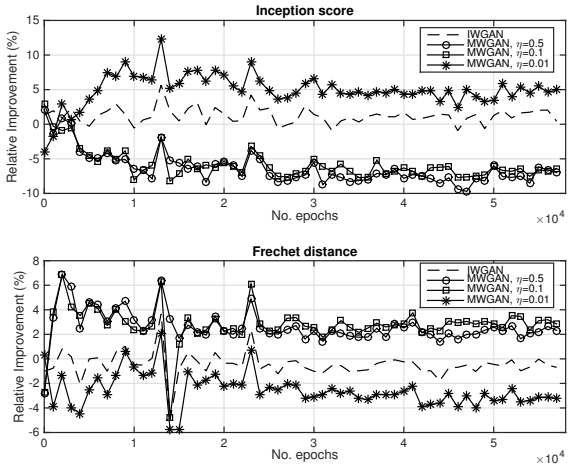

Fig. 4. Similar to Fig. 3 but for the CIFAR-10 dataset. Improvements still happen but they are less prominent while the performance metrics unfortunately produce inconsistent results.

5. CONCLUSIONS AND FUTURE DIRECTIONS

Inspired by the *multiplicative weight update method*, we proposed a novel algorithm to train GANs. Results indicated that the performance is improved when compared to the baseline training procedure. Moreover, WeGAN is not restricted to a particular type of GAN but it can be easily applied to any type. As future directions we list a more extensive study in terms of applications and network architectures, a systematic evaluation of the hyper-parameter's behavior as well as extensions towards adding suitable weights to the Discriminator, too.

6. ACKNOWLEDGEMENTS

The second author is supported by the EU's H2020 research and innovation programme under the MSCA GA 67532 and

the third author is supported by the Stavros Niarchos-FORTH post-doc fellowship for project ARCHERS. We would like to also thank Yannis Sfakianakis for his help in some experimental simulations.

7. REFERENCES

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Proceedings of Annual Conference on Neural Information Processing Systems (NIPS '14)*, 2014, pp. 2672–2680.

[2] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[3] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[4] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," *arXiv preprint arXiv:1610.09585*, 2016.

[5] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[6] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar, et al., "The malicious use of artificial intelligence: Forecasting, prevention, and mitigation," *arXiv preprint arXiv:1802.07228*, 2018.

[7] S. Pascual, A. Bonafonte, and J. Serra, "SEGAN: Speech enhancement generative adversarial network," *arXiv preprint arXiv:1703.09452*, 2017.

[8] Y. Saito, S. Takamichi, and H. Saruwatari, "Statistical parametric speech synthesis incorporating generative adversarial networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 1, pp. 84–96, 2018.

[9] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio, "Maximum-likelihood augmented discrete generative adversarial networks," *arXiv preprint arXiv:1702.07983*, 2017.

[10] K. Schawinski, C. Zhang, H. Zhang, L. Fowler, and G. K. Santhanam, "Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit," *Monthly Notices of the Royal Astronomical Society: Letters*, vol. 467, no. 1, pp. L110–L114, 2017.

[11] M. J. Osborne and A. Rubinstein, *A course in game theory*, MIT press, 1994.

[12] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[14] S. Nowozin, B. Cseke, and R. Tomioka, "f-GAN: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems*, 2016, pp. 271–279.

[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.

[16] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[17] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: a meta-algorithm and applications," *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.

[18] R. D. Hjelm, A. P. Jacob, T. Che, A. Trischler, K. Cho, and Y. Bengio, "Boundary-seeking generative adversarial networks," *arXiv preprint arXiv:1702.08431*, 2017.

[19] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing, "On unifying deep generative models," *arXiv preprint arXiv:1706.00550*, 2017.

[20] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.

[21] Y. LeCun, "The MNIST database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[24] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Citeseer, 2009.