1 The TraVaG System

This supplementary material provides our detailed TraVaG network training algorithms for both autoencoder and GAN. Moreover, we briefly introduce basic computational complexity considerations and show our experimental network tuning procedure and configuration.

1.1 Training Algorithms

The different procedure steps of the TraVaG training together with their relevant parameters are introduced in Algorithm 1.

Algorithm 1: TraVaG Training Procedure

Input: Event log L, number of variants n in L, number of cases m in L, variant binary encoder *VarEnc* : $B(\mathcal{A}^*) \rightarrow [0,1]^{m\times n}$, variant binary decoder *VarDec* : $[0,1]^{m\times n} \rightarrow B(\mathcal{A}^*)$, latent space dimension d, encoder network $enc_\phi : \mathbb{R}^n \rightarrow \mathbb{R}^d$ (parameterized by ϕ), decoder network $dec_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^n$ (parameterized by θ), generator noise space Z with distribution $P(Z)$, generator network $gen_\xi : 2^Z \rightarrow \mathbb{R}^d$ (parameterized by ξ), discriminator network $dis_w : \mathbb{R}^n \rightarrow \{0,1\}$ (parameterized by w), autoencoder loss L_a, generator loss L_G, discriminator loss L_D, autoencoder optimizer O_a, generator optimizer O_G, discriminator optimizer O_D, autoencoder iterations I_a, generator iterations I_G, discriminator iterations i_D per generator step, separate learning rates (η_a, η_G, η_D), autoencoder batch sampling rate q_a, generator batch size b_G, discriminator batch sampling rate q_D, privacy parameters for autoencoder a and discriminator D (noise multiplier Φ_a, Φ_D, clipping norm C_a, C_D, microbatch size S_a, S_D)

Output: Trained autoencoder: enc_ϕ, dec_θ, generator: gen_ξ, discriminator: dis_w parameterized by their weight tensors ϕ, θ, ξ, w

function TRAIN_TraVaG:
- $D = VarEnc(L)$ `// one-hot-encode variants`
- initialize weight tensors ϕ, θ, ξ, w
- **forall** $l \in \{1, 2, ..., I_a - 1, I_a\}$ **do**
  - $\phi, \theta =$ TRAIN_auto$(D, d, enc_\phi, dec_\theta, L_a, O_a, \eta_a, q_a, \Phi_a, C_a, S_a)$
- **forall** $j \in \{1, 2, ..., I_G - 1, I_G\}$ **do**
  - **forall** $k \in \{1, 2, ..., i_D - 1, i_D\}$ **do**
    - $w =$ TRAIN_disc$(D, P(Z), dis_w, gen_\xi, dec_\theta, L_D, O_D, \eta_D, q_D, \Phi_D, C_D, S_D)$
  - $\xi =$ TRAIN_gen$(P(Z), gen_\xi, dis_w, dec_\theta, L_G, O_G, \eta_G, b_G)$
- **return** *TraVaG model* $(enc_\phi, dec_\theta, gen_\xi, dis_w)$

First, the simple event log L is binary-encoded by the one-hot-encoder $VarEnc(\cdot)$. In parallel, all network weight tensors ϕ, θ, ξ, w (for encoder enc_ϕ, decoder dec_θ, generator gen_ξ and discriminator dis_w respectively) are initialized. Depending on the type of activation function used within the respective networks, we either employ He- (*ReLu* activation) or Glorot initialization methods (*Sigmoid, Tanh* activation) [1, 2]. Next, the autoencoder is trained for I_a iterations, i.e. I_a data batches ($TRAIN\_auto$ function). Since the enc_ϕ

and dec_θ networks are utilized within the GAN, this step has to be finished before tuning the generative model. Eventually, equipped with the fixed autoencoder, discriminator dis_w and generator gen_ξ are trained in alternation on the same data D as enc_ϕ and dec_θ ($TRAIN\_disc$ and $TRAIN\_gen$ functions). As proposed by [3], we optimize dis_w for i_D iterations during one step of gen_ξ.[1] While one component is being trained, the other component remains unaltered and vice versa.

The differentially private autoencoder training routine ($TRAIN\_auto$) is illustrated separately in Algorithm 2. At the start, we create a new batch B by sampling data from input D. There are three prominent sampling strategies within the context of DP machine learning [4]. The first method selects each instance independently with a fixed probability. This procedure was investigated in the popular work [5, 6] to derive so-called subsample moment accountants and RDP composition for privacy control. A second approach samples fixed-size data subsets uniformly at random. However, research [7] has shown that this method is frequently outperformed in terms of RDP guarantees by strategy one. Last but not least, the third idea shuffles a dataset uniformly at random, followed by selecting the first x records as the batch. Unfortunately, at the time of this thesis, no proven privacy composition model is known to incorporate the shuffled data selection. For our work, we choose method one due to the broadly researched characteristics and superior anonymization performance (specified by selection rate q_a).

Algorithm 2: Autoencoder Training Subroutine

Input: One-hot-encoded variants D, latent space dimension d, encoder network $enc_\phi : \mathbb{R}^n \to \mathbb{R}^d$ (parameterized by ϕ), decoder network $dec_\theta : \mathbb{R}^d \to \mathbb{R}^n$ (parameterized by θ), loss function L_a, optimizer method O_a, learning rate η_a, batch sampling rate q_a, noise multiplier Φ_a, clipping norm C_a, microbatch size S_a)

Output: Trained autoencoder: enc_ϕ, dec_θ and the weight tensors ϕ, θ

function TRAIN_auto($D, d, enc_\phi, dec_\theta, L_a, O_a, \eta_a, q_a, \Phi_a, C_a, S_a$)
- randomly sample data batch B from D by rate q_a
- $k = \lfloor m \cdot q_a / S_a \rfloor$ `// estimate number of microbatches`
- partition B into $B_1, \ldots, B_k$ each of size S_a `// create microbatches`
- **forall** $i \in \{1, 2, ..., k-1, k\}$ **do**
  - $g^i_\phi = \nabla_\phi(L_a(B_i, dec_\theta(enc_\phi(B_i))))$ `// backprobagation for` ϕ
  - $g^i_\theta = \nabla_\theta(L_a(B_i, dec_\theta(enc_\phi(B_i))))$ `// backpropagation for` θ
- $g_\phi = \frac{1}{k} \sum_{i=1}^k g^i_\phi$ `// average gradients over microbatches`
- $g_\theta = \frac{1}{k}((\sum_{i=1}^k \text{clip}(g^i_\theta, C_a)) + \mathcal{N}(0, \Phi_a^2 C_a^2 I))$ `// noise addition`
- $\phi, \theta = O_a(\phi, \theta, g_\phi, g_\theta, \eta_a)$ `// run optimizer method`
- **return** ϕ, θ

As highlighted in our main paper, we moreover provide the feature of further partitioning B into k microbatches $B_1, \ldots, B_k$ of size S_a. For all of these microbatches, standard backpropagation is conducted to obtain the individual encoder and decoder gradients g^i_ϕ and g^i_θ over the loss function $L_a(B_i, dec_\theta(enc_\phi(B_i)))$ ($i = 1 \ldots k$). We base this op-

[1] In total we thus run I_G iterations on the generator and $i_D \cdot I_G$ iterations on the discriminator.

eration on the underlying idea that an autoencoder pipeline works sufficiently once the distance between its input x and the decoded result of the encoded input $dec_\theta(enc_\phi(x))$ becomes negligible. As a suitable distance function on high-dimensional binary x, y it is recommended to integrate the binary cross entropy loss:

$$\operatorname{dist}(x,y) = -\sum_{j=1}^{n} y_{(j)}\log(x_{(j)}) - \sum_{j=1}^{n}(1-y_{(j)})\log(1-x_{(j)}) \tag{1}$$

where $x_{(j)}$ denotes the jth coordinate of input $x \in [0,1]^n$ [8, 9]. Considering our sample micrcobatch $B_i = \{x_j\}_{j=1}^{S_a}$, we thus define the complete autoencoder loss $L_a(\cdot)$ as follows:

$$L_a(B_i, dec_\theta(enc_\phi(B_i))) \coloneqq \sum_{j=1}^{S_a} \operatorname{dist}(x_j, dec_\theta(enc_\phi(x_j))). \tag{2}$$

After both microbatch gradient stacks g_ϕ^i and g_θ^i $(i = 1 \dots k)$ are computed, they are averaged in different ways, depending on the specific network component. Since the encoder does not participate in the process of training the GAN or synthesizing new event data, it does not need to be optimized privately [10, 11, 12]. As a result, we obtain the aggregated gradient g_ϕ by a simple mean calculation. On the contrary, the decoder is strongly involved in the anonymization process and released to the public which is why g_θ represents a mean of the clipped microbatch gradients supplemented with Gaussian noise. Finally, new parameter updates ϕ, θ are found by a gradient descent optimization scheme O_a that reduce the autoencoder error denoted by loss L_a.

Similar to the autoencoder training, we present the differentially private discriminator training procedure $TRAIN\_disc$ as Algorithm 3. In the beginning we again extract a randomly sampled batch B from D by rate q_D, that is then further splitted into k microbatches $B_1, \dots, B_k$ (see Algorithm 2). Since the discriminator has the goal of learning to differentiate between real and fake, generated data, we additionally create k S_D-sized batches $\hat{B}$ of synthesized binary variants. For this purpose, first, i.i.d random noise $\{z_j\}_{j=1}^{S_D}$ is drawn from a Gaussian distribution $P(Z)$ of user-defined dimensionality.[2] Next, the $\hat{B}$ data are derived by feeding the noise into the generator gen_ξ, followed by the decoder dec_θ. Considering these two types of batches B_i and $\hat{B}$, the discriminator goal can be formally defined as maximizing the probability of a correct guess, i.e. $\mathbb{E}_{z\sim Z}[1 - dis_w(dec_\theta(gen_\xi(z)))]$ in case of synthetic inputs (noise z resulting in $\hat{B}$) and $\mathbb{E}_{x\sim X}[dis_w(x)]$ in case of real inputs ($x \in B_i$, following data distribution X). For a smoother training experience, we extend the typical binary output of dis_w to a continuous range $[0, 1]$, indicating the confidence of the respective decision [13]. A natural penalty function for such discriminatory components is the zero-sum objective loss function in Equation 3 which is motivated by the so-called Wasserstein distance between two distributions [14, 9].

[2]Note that the noise may also follow different probability distributions and we thus keep the random variable Z more general [3].

Algorithm 3: Discriminator Training Subroutine

Input: One-hot-encoded variants D, generator noise distribution $P(Z)$, discriminator network $dis_w : \mathbb{R}^n \to [0,1]$ (parameterized by w), generator network $gen_\xi : 2^Z \to \mathbb{R}^d$ (parameterized by ξ), decoder network $dec_\theta : \mathbb{R}^d \to \mathbb{R}^n$ (parameterized by θ), optimizer method O_D, learning rate η_D, batch sampling rate q_D, noise multiplier Φ_D, clipping norm C_D, microbatch size S_D)

Output: Trained discriminator: dis_w and the weight tensor w

function TRAIN_disc($D, P(Z), dis_w, gen_\xi, dec_\theta, O_D, \eta_D, q_D, \Phi_D, C_D, S_D$)
- randomly sample data batch B from D by rate q_D
- $k = \lfloor m \cdot q_D / S_D \rfloor$ `// estimate number of microbatches`
- partition B into $B_1, \ldots, B_k$ each of size S_D `// create microbatches`
- **forall** $i \in \{1, 2, ..., k-1, k\}$ **do**
  - sample i.i.d noise batch $\{z_j\}_{j=1}^{S_D} \sim Z$ from $P(Z)$
  - $\hat{B} = \{dec_\theta(gen_\xi(z_j))\}_{j=1}^{S_D}$ `// transform noise batch`
  - $L_D(\hat{B}, B_i) \coloneqq \frac{1}{S_D} \sum_{b \in B_i} dis_w(b) - \frac{1}{S_D} \sum_{b \in \hat{B}} dis_w(b)$ `// WGAN loss`
  - $g_w^i = \nabla_w (L_D(\hat{B}, B_i))$ `// backprobagation for` w
- $g_w = \frac{1}{k}((\sum_{i=1}^k \text{clip}(g_w^i, C_D)) + \mathcal{N}(0, \Phi_D^2 C_D^2 I))$ `// noise addition`
- $w = O_D(w, g_w, \eta_D)$ `// run optimizer method`
- **return** w

$$L_D(\hat{B}, B_i) \coloneqq \frac{1}{S_D} \sum_{b \in B_i} dis_w(b) - \frac{1}{S_D} \sum_{b \in \hat{B}} dis_w(b) \tag{3}$$

Here, the first part describes the average positive response of dis_w to real data points and the second part the average negative feedback of dis_w to synthetic data points. We compute L_D for all microbatch pairs $\hat{B}, B_i$ ($i = 1 \ldots k$) and derive the corresponding gradients g_w^i with respect to parameter w by standard backpropagation techniques. As for the autoencoder network dec_θ, the individual g_w^i information are then differentially privately aggregated to g_w with a clipping norm and Gaussian noise addition. This operation is necessary since the discriminator receives confidential original data and thus needs to be trained anonymously [11, 9]. Eventually, we obtain new improved network weights w by executing the gradient descent optimizer O_D with the old w, g_w and the learning rate η_D.

Last but not least, Algorithm 4 shows the non-privatized generator training routine $TRAIN\_gen$. In contrast to the discriminator, we only require a batch of synthetic generated binary data B with size b_G, but no access to the real source D. To construct B, again b_G i.i.d noise samples are drawn from a Gaussian distribution at random and sent to the generator gen_ξ, followed by the decoder dec_θ. We recall that the overall goal of the generator is to trick the discriminator in a way that any fake data of B is labeled as original. Hence, it formally attempts to minimize the probability of dis_w making a correct guess, i.e. $\mathbb{E}_{z \sim Z}[1 - dis_w(dec_\theta(gen_\xi(z)))]$. An approximate penalty function re-

Algorithm 4: Generator Training Subroutine

Input: Generator noise distribution $P(Z)$, generator network $gen_\xi : 2^Z \to \mathbb{R}^d$ (parameterized by ξ), discriminator network $dis_w : \mathbb{R}^n \to \{0, 1\}$ (parameterized by w), decoder network $dec_\theta : \mathbb{R}^d \to \mathbb{R}^n$ (parameterized by θ), optimizer method O_G, learning rate η_G, batch size b_G

Output: Trained generator: gen_ξ and the weight tensor ξ

function TRAIN_gen$(P(Z), gen_\xi, dis_w, dec_\theta, O_G, \eta_G, b_G)$
- sample i.i.d noise batch $\{z_i\}_{i=1}^{b_G} \sim Z$ from $P(Z)$
- $B = \{dec_\theta(gen_\xi(z_i))\}_{i=1}^{b_G}$ `// transform noise batch`
- $L_G(B) \coloneqq -\frac{1}{b_G} \sum_{b \in B} dis_w(b)$ `// discriminator WGAN loss function`
- $g_\xi = \nabla_\xi(L_G(B))$ `// backpropagation for` ξ
- $\xi = O_G(\xi, g_\xi, \eta_G)$ `// run optimizer method`
- **return** ξ

flecting this principle is the negative average discriminator response over B in Eq. 4 that represents the counterpart of Eq. 3 [13]. For our generator we employ Eq. 4 as the loss L_G.

$$L_G(B) \coloneqq -\frac{1}{b_G} \sum_{b \in B} dis_w(b) \tag{4}$$

At last, the gradient g_ξ of the computed L_G with respect to ξ is determined and forwarded to a standard non-DP gradient descent method O_G to find new loss-minimizing parameters ξ. As the generator does not access nor release or process any original event data[3], this optimization step is non-privatized and therefore equal to optimization implemented in traditional non-DP GAN generators [3, 9].

After the successful two-phase training of autoencoder and GAN, the tuned decoder dec_θ and generator gen_ξ are released to the public for application. Starting from a Gaussian random sample z, a synthesized variant σ can be obtained by calculating

$$\sigma = VarDec(dec_\theta(gen_\xi(z))) \tag{5}$$

where $VarDec$ denotes the one-hot decoder (see Algorithm 1). As previously mentioned, σ will always represent a true variant due to the equality of feature-space and variant universe. Consequently, the full power of TraVaG is revealed once we repeat this procedure multiple times. The more synthetic data σ is created, the better the consolidated TraVaG output, i.e. the different new anonymized variants approximate the original variant distribution. However, we note that this process does not converge to the true variant frequencies, but to the TraVaG-internal learned anonymous version thereof.[4]

[3]In fact, the generator only handles data indirectly through the differentially private discriminator- and decoder outputs. As a result, it also maintains DP via the post-processing immunity property.

[4]It is hence recommended to run TraVaG at least as often as the number of cases in the original event

1.2 TraVaG Configuration and Network Settings

All TraVaG artificial neural networks are configured by a semi-automated tuning approach with respect to the different input datasets. Whereas most design decisions and hyperparameters are tweaked according to results of manual tests as well as research experience, the settings: *batch size*, *number of iterations* and *noise multiplier* are automatically optimized via a grid-search [15] at fixed privacy levels. Subsequently, we present the most important aspects.

- The encoder and decoder network architectures are both two-layer perceptrons with Tanh activation functions except for the decoder output that is created by a Sigmoid function [16]. The inner hidden dimension is automatically selected as the mean of input dimension n and latent space dimension d. We define d dependent on the input data: 128 for both logs Sepsis and BPIC-2013.

- The discriminator network consists of 3 linear feed-forward layers that gradually reduce their dimensions until the last one-dimensional output is obtained. At the first step, the input size n is reduced by $2n/3$; next, we decrease $2n/3$ to $n/3$. In between the layers, leaky ReLu functions with slope 0.3 are added to account for the sparse gradient problem [16, 13].

- On the contrary, our generator differs from the other architectures in that it represents a 3 layer *ResNet* model with Tanh activation [17]. Instead of the simple feed-forward procedure, the output of one block is additionally added to the next block result to improve the backpropagation process. We choose the Gaussian noise space dimension dependent on the data to be 128 for both Sepsis and BPIC-2013 logs.

- For the autoencoder, the Adam optimizer [18] is employed with a learning rate of 0.005 and exponential decay parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. In direct competition to RMSProp, these settings yielded a faster convergence for all event logs and privacy settings.

- Conversely, our GAN (both generator and discriminator) is optimized with RMSProb [19] due to superior test performance on Sepsis and BPIC-2013 data. We again select a learning rate of 0.005 and a smoothing constant $\alpha = 0.99$ independent of the input. Moreover, the number of discriminator optimizations per generator step is set to $i_D = 15$.

- The gradient clipping norm is chosen to be the average median L^2-norm observed during non-private training loops with all event logs (0.012 for the autoencoder, 0.022 for generator and discriminator). As a result, gradient updates can still propagate through the networks at a sufficient magnitude.

- For the noise multiplier Φ, the training iterations I and the batch size b we discover the strongest influence on both TraVaG convergence and privacy levels. Therefore,

log. In case smaller privatized datasets are needed, the output can be downsampled during postprocessing rounds.

once the other parameters are set, an automated grid-search optimization [15] is started as follows. While keeping the remaining configuration fixed, we first determine combinations of Φ, I and b that lead to the same (ϵ, δ) budget (see [6]). Then, TraVaG is trained for all of these parameter triples to find the performance-optimal model. Here, performance is evaluated by means of data utility metrics and convergence speed.

Note that the afore-mentioned TraVaG tuning process is conducted in a lab environment without accounting explicit privacy cost when repeatedly querying data for training. We justify this aspect by our goal of comparing the generative framework with TraVaS and the prefix-based method under equally optimal conditions. In case no such confidential hyperparameter optimization is possible, e.g., at industrial applications, we refer to studied strategies for deriving high-performance machine learning models without violating privacy guarantees [20, 21, 22].

1.3 Note of Authorship

This document is part of the supplementary material created for the paper *Releasing Differentially Private Event Logs Using Generative Models*, written by Frederik Wangelik, Majid Rafiei, Mahsa Pourbafrani and Wil M.P. Van der Aalst. Please contact *frederik.wangelik@rwth-aachen.de* for further information.

References

[1] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, ser. JMLR Proceedings, Y. W. Teh and D. M. Titterington, Eds., vol. 9. JMLR.org, 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 1026–1034. [Online]. Available: https://doi.org/10.1109/ICCV.2015.123

[3] 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 *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 2672–2680. [Online]. Available: https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

[4] H. B. McMahan and G. Andrew, “A general approach to adding differential privacy to iterative training procedures,” *CoRR*, vol. abs/1812.06210, 2018. [Online]. Available: http://arxiv.org/abs/1812.06210

[5] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 308–318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[6] I. Mironov, “Rényi differential privacy,” in *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*. IEEE Computer Society, 2017, pp. 263–275. [Online]. Available: https://doi.org/10.1109/CSF.2017.11

[7] Y. Wang, B. Balle, and S. P. Kasiviswanathan, “Subsampled rényi differential privacy and analytical moments accountant,” *J. Priv. Confidentiality*, vol. 10, no. 2, 2020. [Online]. Available: https://doi.org/10.29012/jpc.723

[8] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6114

[9] U. T. Tantipongpipat, C. Waites, D. Boob, A. A. Siva, and R. Cummings, "Differentially private synthetic mixed-type data generation for unsupervised learning," *Intell. Decis. Technol.*, vol. 15, no. 4, pp. 779–807, 2021. [Online]. Available: https://doi.org/10.3233/IDT-210195

[10] G. Ács, L. Melis, C. Castelluccia, and E. D. Cristofaro, "Differentially private mixture of generative neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 6, pp. 1109–1121, 2019. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2855136

[11] N. C. Abay, Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and L. Sweeney, "Privacy preserving synthetic data release using deep learning," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part I*, ser. Lecture Notes in Computer Science, M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim, Eds., vol. 11051. Springer, 2018, pp. 510–526. [Online]. Available: https://doi.org/10.1007/978-3-030-10925-7_31

[12] Q. Chen, C. Xiang, M. Xue, B. Li, N. Borisov, D. Kaafar, and H. Zhu, "Differentially private data generative models," *CoRR*, vol. abs/1812.02274, 2018. [Online]. Available: http://arxiv.org/abs/1812.02274

[13] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5767–5777. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/892c3b1c6dccd52936e27cbd0ff683d6-Abstract.html

[14] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *CoRR*, vol. abs/1701.07875, 2017. [Online]. Available: http://arxiv.org/abs/1701.07875

[15] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for NAS," *CoRR*, vol. abs/1912.06059, 2019. [Online]. Available: http://arxiv.org/abs/1912.06059

[16] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022. [Online]. Available: https://doi.org/10.1016/j.neucom.2022.06.111

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.* IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA,*

USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[19] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[20] K. Chaudhuri and S. A. Vinterbo, "A stability-based validation procedure for differentially private machine learning," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 2652–2660. [Online]. Available: https://proceedings.neurips.cc/paper/2013/hash/e6d8545daa42d5ced125a4bf747b3688-Abstract.html

[21] J. Liu and K. Talwar, "Private selection from private candidates," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, M. Charikar and E. Cohen, Eds. ACM, 2019, pp. 298–309. [Online]. Available: https://doi.org/10.1145/3313276.3316377

[22] B. K. Beaulieu-Jones, Z. S. Wu, C. Williams, R. Lee, S. P. Bhavnani, J. B. Byrd, and C. S. Greene, "Privacy-preserving generative deep neural networks support clinical data sharing," *Circulation: Cardiovascular Quality and Outcomes*, vol. 12, no. 7, p. e005122, 2019. [Online]. Available: https://www.ahajournals.org/doi/abs/10.1161/CIRCOUTCOMES.118.005122