Double Oracle Neural Architecture Search for Game Theoretic Deep Learning Models

Aye Phyu Phyu Aung, Xinrun Wang, Ruiyu Wang, *Graduate Student Member, IEEE*, Hau Chan, Bo An, *Senior Member, IEEE*, Xiaoli Li, *Fellow, IEEE*, and J. Senthilnath, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a new approach to train deep learning models using game theory concepts including Generative Adversarial Networks (GANs) and Adversarial Training (AT) where we deploy a double-oracle framework using best response oracles. GAN is essentially a two-player zero-sum game between the generator and the discriminator. The same concept can be applied to AT with attacker and classifier as players. Training these models is challenging as a pure Nash equilibrium may not exist and even finding the mixed Nash equilibrium is difficult as training algorithms for both GAN and AT have a large-scale strategy space. Extending our preliminary model DO-GAN, we propose the methods to apply the double oracle framework concept to Adversarial Neural Architecture Search (NAS for GAN) and Adversarial Training (NAS for AT) algorithms. We first generalize the players' strategies as the trained models of generator and discriminator from the best response oracles. We then compute the meta-strategies using a linear program. For scalability of the framework where multiple network models of best responses are stored in the memory, we prune the weakly-dominated players' strategies to keep the oracles from becoming intractable. Finally, we conduct experiments on MNIST, CIFAR-10 and TinyImageNet for DONAS-GAN. We also evaluate the robustness under FGSM and PGD attacks on CIFAR-10, SVHN and TinyImageNet for DONAS-AT. We show that all our variants have significant improvements in both subjective qualitative evaluation and quantitative metrics, compared with their respective base architectures.**

***Index Terms*—Generative adversarial networks, neural architecture search, adversarial training, double oracle, game theory.**

Received 4 April 2024; revised 27 September 2024 and 16 February 2025; accepted 30 March 2025. Date of publication 11 April 2025; date of current version 30 April 2025. An earlier version of this paper was presented at CVPR 2022. The associate editor coordinating the review of this article and approving it for publication was Dr. Jianwen Xie. *(Aye Phyu Phyu Aung and Xinrun Wang contributed equally to this work.) (Corresponding authors: Aye Phyu Phyu Aung; Xinrun Wang; J. Senthilnath.)*

Aye Phyu Phyu Aung and J. Senthilnath are with the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore 138632 (e-mail: aye_phyu_phyu_aung@i2r.a-star.edu.sg; J_Senthilnath@i2r.a-star.edu.sg).

Xinrun Wang is with the School of Computing and Information Systems, Singapore Management University, Singapore 188065 (e-mail: xrwang@smu.edu.sg).

Ruiyu Wang is with the Division of Robotics, Perception and Learning, KTH Royal Institute of Technology, 100 44 Stockholm, Sweden (e-mail: ruiyuw@kth.se).

Hau Chan is with the School of Computing, University of Nebraska–Lincoln, Lincoln, NE 68588 USA (e-mail: hchan3@unl.edu).

Bo An is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: boan@ntu.edu.sg).

Xiaoli Li is with the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore 138632, also with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, and also with the A*STAR Centre for Frontier AI Research, Singapore 138632 (e-mail: xlli@i2r.a-star.edu.sg).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIP.2025.3558420, provided by the authors.

Digital Object Identifier 10.1109/TIP.2025.3558420

I. Introduction

MOST machine learning algorithms involve optimizing a single set of parameters to minimize a single cost function. For some deep learning models like GAN and AT, however, two or more "players" adapt their own parameters to minimize their own cost, in competition with the other players like a game. There are two most popular architectures that apply game theory concepts in deep learning models: i) generative adversarial networks (GAN) where a generator network generates images, while the discriminator distinguishes the generated images from real images, resulting in the ability of the generator to produce realistic images [5], [11], [18], and ii) adversarial training (AT) where a classifier is trained against an attacker who can manipulate the inputs to decrease the performance of the classifier [12], [25]. In both cases, the training algorithms can be viewed as a two-player zero-sum game where each player is alternately trained to maximize their respective utilities until convergence corresponding to a Nash Equilibrium (NE) [1].

However, pure NE cannot be reliably reached by existing algorithms as pure NE may not exist [7], [27]. This also leads to unstable training in GANs depending on the data and the hyperparameters. In this case, mixed NE is a more suitable solution concept [16]. Several recent works propose mixture architectures with multiple generators and discriminators that consider mixed NE such as MIX+GAN [1] and MGAN [15] but theoretically cannot guarantee to converge to mixed NE. Mirror-GAN [16] computes the mixed NE by sampling over the infinite-dimensional strategy space and proposes provably convergent proximal methods. However, the sampling approach may not be efficient as mixed NE may only have a few strategies in the support set.

On the other hand, Neural Architecture Search (NAS) approaches have also shown promising results. They have been applied to some computer vision tasks, such as image classification, dense image prediction and object detection. Recently, NAS has been researched in GANs [10], [29]. Particularly, Adversarial-NAS [8] searches both of the generator and discriminator simultaneously in a differentiable manner using gradient-based method NAS and a large architecture search space. Moreover, NAS is also used to search the classifier architectures in Adversarial Training (AT) for robustness. AdvRush [28] considers both transparent-box attacks and opaque-box attacks using the input loss landscape of the networks to represent their intrinsic robustness. However, the NAS techniques only derive one final architecture with a criterion such as maximum weight from the search space,

ignoring other top-performing results from the neural architecture search.

Double Oracle (DO) algorithm [26] is a powerful framework to compute mixed NE in large-scale games. The algorithm starts with a restricted game that is initialized with a small set of actions and solves it to get the NE strategies of the restricted game. The algorithm then computes players' best responses using oracles to the NE strategies and adds them into the restricted game for the next iteration. DO framework has been applied in various disciplines [3], [17], as well as Multi-agent Reinforcement Learning (MARL) [21]. Our previous work DO-GAN [2] train GANs by deploying a double oracle framework using generator and discriminator from the best response oracles.

Extending the successful applications of the DO framework on GAN, we propose a Double Oracle Neural Architecture Search for GAN and AT (DONAS-GAN and DONAS-AT) to show the improvement of performance by the double oracle framework on neural architecture search for the two game theoretic deep learning models. This paper presents four key contributions.

1) We propose the general double oracle framework for Neural Architecture Search (DONAS) with two main components of GAN and AT (generator/discriminator or classifier/attacker) as players. The players in DONAS obtain the best responses from their oracles and add the utilities to a meta-matrix. We use a linear program to obtain the probability distributions of the players' pure strategies (meta-strategies) for the respective oracles and pruning for scalable implementation of the extended algorithms.
2) We propose DONAS-GAN variant with the respective oracles to search multiple generators and discriminators referencing the techniques from Adversarial-NAS.
3) We also propose DONAS-AT variant with the classifier/attacker oracles for robustness in the classification tasks referencing the techniques from AdvRush to search the classifier model and applying the general double oracle framework.
4) We propose two methods: (a): Harmonic Mean and (b): Nash, to sequentially finetune the searched networks generators and discriminators for DONAS-GAN as well as classifiers for DONAS-AT.

Finally, we provide a comprehensive evaluation on the performances of DONAS-GAN and DONAS-AT using real-world datasets. Experiment results show that our variants have significant improvements in terms of both subjective qualitative evaluation and quantitative metrics i.e., FID score.

II. Related Works

In this section, we briefly introduce existing game theoretic deep learning architectures, double oracle algorithm and its applications that are related to our work.

A. GAN, at and NAS

1) Generative Adversarial Networks (GAN) Architectures: Various GAN architectures have been proposed to improve their performance. Apart from single architecture advancements, multi-model architectures have also shown promising improvements to the GAN training process. Considering mixed NE, MIX+GAN [1] maintains a mixture of generators and discriminators with the same network architecture but has its own trainable parameters. However, training a mixture of networks without parameter sharing makes the algorithm computationally expensive. Mixture Generative Adversarial Nets (MGAN) [15] proposes to capture diverse data modes by formulating GAN as a game between a classifier, a discriminator and multiple generators with parameter sharing. However, MIX+GAN and MGAN cannot converge to mixed NE. Mirror-GAN [16] finds the mixed NE by sampling over the infinite-dimensional strategy space and proposes provably convergent proximal methods. This approach may be inefficient to compute mixed NE as the mixed NE may only have a few strategies with positive probabilities in the infinite strategy space. [8]

2) Adversarial Training (at): As the NAS becomes widely researched, investigations on its intrinsic vulnerability and enhancement of robustness have also increased. Numerous approaches have been proposed but Adversarial Training (AT) based methods [25] still remain the strongest defense approach. RobNet [13], randomly samples architectures from a search space and adversarially trains each of them. However, this is very computationally expensive. Moreover, evolutionary algorithm architectures such as RoCo-NAS [9] are restricted to only opaque-box attacks.

3) Neural Architecture Search (NAS): With Automatic Machine Learning (AutoML), Neural Architecture Search (NAS) has become one of the most important directions for machine learning. In the context of generative networks, AutoGAN [10] and Adversarial-NAS [8] search the generator and discriminator architectures simultaneously in a differentiable manner and GA-NAS [29] proposes to use adversarial learning approach where the searched generator is trained by reinforcement learning based on rewards provided by the discriminator. Similarly for AT, AdvRush [28] proposes an adversarial robustness-aware neural architecture search algorithm using the input loss landscape of the neural networks to represent their intrinsic robustness. However, in both GAN and AT, NAS only samples the architectures from the search space by picking the operations with maximum weights. To improve the modes that are missed for training, we propose DONAS for GANs and AT by sampling and finetuning multiple networks instead of just selecting the maximum. DONAS does not induce any restriction on NAS.

B. Double Oracle Algorithm

Double Oracle (DO) algorithm starts with a small restricted game between two players and solves it to get the players' strategies at Nash Equilibrium (NE) of the restricted game. The algorithm then exploits the respective best response oracles for additional strategies of the players. The DO algorithm terminates when the best response utilities are not higher than the equilibrium utility of the current restricted game. Hence, it finds the NE of the game without enumerating the entire strategy space. Moreover, in two-player zero-sum games, DO

TABLE I
COMPARISON OF TERMINOLOGIES BETWEEN GAME THEORY AND GAN

Game Theory terminology	GAN terminology
Player	Generator/discriminator
Strategy	The parameter setting of generator/discriminator, e.g., π_g and π_d
Policy	The sequence of parameters (strategies) till epoch t, e.g., $(\pi_g^1, \pi_g^2, ..., \pi_g^t)$ Note: Not used in this paper.
Game	The minmax game between generator and discriminator
Meta-game/ meta-matrix	The minmax game between generator and discriminator with their respective set of strategies at epoch t of DO framework
Meta-strategy	The mixed NE strategy of generator/discriminator at epoch t

converges to a min-max equilibrium [26]. DO framework is used to solve large-scale normal-form and extensive-form games such as security games [17], [32], poker games [34] and search games [4] and it is widely used in various disciplines. Related to our research, DO is also used to alternately train the multiple generators and discriminators for the different architectures of GANs with best response oracles [2]. We present the corresponding terminologies between GAN and game theory in Table I.

III. PRELIMINARIES

In this section, we mathematically explain the preliminary works to effectively support our approach in addition to our previous work DO-GAN [2].

A. Adversarial Training

Given a fixed classifier with parameters θ, a dataset X (e.g., an image x with true label y), and classification loss $\mathcal{L}$, a bounded non-targeted attack is done by

$$\max_{\delta} \mathcal{L}(x+\delta, y, \theta), \delta \in S, \tag{1}$$

where $S = \{\delta \mid \|\delta\|_p \leq \epsilon\}$ is the space for the perturbation, $\|\cdot\|_p$ is the p-norm distance metric of adversarial perturbation and ϵ is the manipulation budget. We leverage Projected Gradient Descent (PGD) which uses the sign of the gradients and projects the perturbations into the set S, to construct an adversarial example for each iteration t with step size ϵ.

$$x^t = \Pi_{x+S}(x^{t-1} + \epsilon \cdot \text{sgn}(\nabla_x \mathcal{L}(x^{t-1}, y, \theta))). \tag{2}$$

K-PGD [25] is the iterative attack of the adversarial examples constructed in Eq. (2) with uniform random noise as initialization, i.e., $x^0 = x + \zeta$ where ζ is from noise distribution. The strength of PGD attacks to generate adversarial examples depends on the number of iterations. It has an inner loop that constructs adversarial examples, while its outer loop updates the model using mini-batch stochastic gradient descent on the generated examples. However, it is generally slow. Thus, we adapt Free AT [30] as the AT algorithm of our DO-Framework. It computes the ascent step by re-using the backward pass needed for the descent step. To allow for multiple adversarial updates to the same image without performing multiple backward passes, Free AT trains on the same mini-batch several times in a row.

B. Double Oracle Framework for GANs

DO-GAN [2] translates GAN as a two-player zero-sum game between the generator player and the discriminator player. A two-player zero sum game is defined as a tuple (Π, U) where $\Pi = (\Pi_g, \Pi_d)$ is the set of strategies for generator player and discriminator player, $U : \Pi_g \times \Pi_d \rightarrow R$ is a utility payoff table for each joint policy played by both players. For zero-sum game, the gain of generator player $u(\pi_g, \pi_d)$ is equal to the loss of discriminator player $-u(\pi_g, \pi_d)$.

Then, at each iteration t, DO-GAN creates a restricted meta-matrix game with the newly trained generators π_g' and discriminators π_d' as strategies of the two players, calculates the payoffs, and expected utilities of mixed strategies to compute mixed NE of the restricted meta-matrix game U^t between generator and discriminator players to get the probability distributions (σ_g^*, σ_d^*). Using the probability distribution, we compute the best responses and adds them into the restricted game for the next iteration. We adapted [21] as the general framework for extending the double oracle framework to NAS for GAN and AT. For the scalable implementation, DO-GAN uses pruning and continual learning techniques and removes the least-contributing strategies toward each player's winning to keep the number of generator and discriminator models from the best response at a size that the memory can support.

IV. DONAS FOR GAME THEORETIC DEEP LEARNING

In this section, we describe the double oracle frameworks with network architecture search for GAN and AT consisting of extending DO-GAN. Moreover, recent works propose evolutionary computation methods to neural architecture search methods for more compact and flexible architectures [35], [36]. Inspired by the evolutionary methods, we propose Adversarial-Neural Architecture Search (DONAS-GAN) and Neural Architecture Search for Adversarial Training (DONAS-AT) to compute the mixed NE efficiently improving the generalizability and flexibility in model architecture. Over the different NAS methods for GAN and AT, the DO framework builds a restricted meta-matrix game between the two players and computes the mixed NE of the meta-matrix game, then iteratively adds more strategies of the two players (generators/discriminators or classifier/attacker) into the meta-matrix game until termination.

Algorithm 1 DONAS: General Framework

1: P_1, P_2 = `InitializeModels`()
2: $\mathcal{P}_1 \leftarrow \{P_1\}$, $\mathcal{P}_2 \leftarrow \{P_2\}$, $\sigma_{p1}^{1*} = \sigma_{p2}^{1*} = [1]$
3: **for** epoch $t = 1, 2, \ldots$ **do**
4: P_1 = `Player1_NASSearch`()
5: Π_1 = `SampleFromSupernet`(P_1)
6: P_2 = `Player2_NASSearch`()
7: Π_2 = `SampleFromSupernet`(P_2)
8: $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \Pi_1$, $\mathcal{P}_2 \leftarrow \mathcal{P}_2 \cup \Pi_2$
9: `FineTuning`()
10: Augment U^{t-1} with P_1 and P_2 to obtain U^t
11: Compute mixed-NE $\langle \sigma_{p1}^{t*}, \sigma_{p2}^{t*} \rangle$ for U^t
12: **if** `TerminationCheck`($U^t, \sigma_{p1}^{t*}, \sigma_{p2}^{t*}$) **then**
13: **break**
14: **end if**
15: **end for**

A. General Framework for DONAS

We have discussed the two prominent approaches to optimizing network architecture in adversarial machine learning, mainly, DO as the mixed-architecture approach, and Adversarial-NAS and AdvRush as the NAS approach for GANs and AT, respectively. DO finds the best response architecture by only updating weights for predefined architecture in the oracles, while NAS methods select only one final network from the search space and may miss the information trained by other top search architectures. To mitigate the issues in both methods, we propose the DONAS approach to combine them, allowing searched networks in a mixed-architecture framework.

We describe the general framework for our proposed DONAS. We adapt the double oracle framework, in which Neural Architecture Search is used as the best response oracle for the two players: generator-discriminator for GANs and classifier-attacker for AT. We search and sample the best response strategies for the two players with respective oracles. Then, we finetune to improve the quality of the image generation and robustness against adversarial attacks. To speed up the training, we introduce the terminating condition ϵ.

Algorithm 1 initializes the first pair of models and stores them in the two arrays $\mathcal{P}_1$ and $\mathcal{P}_2$. Then, we can randomly sample pure strategies to augment the meta-game and compute the distribution of the player strategies. We initialize the meta-strategies $\sigma_{p1}^{1*} = [1]$ and $\sigma_{p2}^{1*} = [1]$ since only one pair of player strategies is available (line 1-2). For each epoch, we search the new strategy for each player (line 4, 6), obtaining the supernets P_1 and P_2. Next, we sample the networks to derive the architecture from supernet (line 5, 7). We then finetune the selected models (line 9). In DONAS for AT, we do not have sampling and finetuning for the best response oracle for attacker player since we do not need to search architecture for the attacker player. Next, we generate the meta-game U^t and compute mixed NE with linear programming proposed in [2] to obtain σ_{p1}^* and σ_{p2}^* (line 10-11). The algorithm terminates if the terminating condition ϵ is satisfied (line 12) which we follow the procedures of the DO-GAN/P [2].

B. DONAS for GAN

In this section, we use DONAS for GANs to allow the use of multiple generators and discriminators. We let $\mathcal{P}_1 = \mathcal{G}$ and $\mathcal{P}_2 = \mathcal{D}$ as the two-player zero-sum game between the generator and discriminator with their architectures as α and β respectively. Theoretically, using multiple generators and discriminators with mixed strategies covers multiple modes to produce better generated images [1], [15], [16].

Algorithm 2 DONAS for GANs

Procedure: `InitializeModels`()
1: Initialize the generator G and discriminator D
2: `FineTune`(`G`)
Procedure: `GeneratorOracle`()
3: G = initialize()
4: **for** s steps **do**
5: Sample mini-batch of $2m$ noise samples.
6: Update the architecture of generator:
$\nabla_\alpha \frac{1}{m} \sum_{i=1}^{m} [\sum_{j=1}^{|\mathcal{D}|} \sigma_d^{j*} \cdot \log(1 - D_j(G(z^i)))]$
7: Update the weights of generator:
$\nabla_{W_G} \frac{1}{m} \sum_{i>m}^{2m} [\sum_{j=1}^{|\mathcal{D}|} \sigma_d^{j*} \cdot \log(1 - D_j(G(z^i)))]$
8: **end for**
Procedure: `DiscriminatorOracle`()
9: D = initialize()
10: **for** s steps **do**
11: Sample $2m$ samples for noise and real data
12: Update the architecture of discriminator:
$\nabla_\beta \frac{1}{m} \sum_{i=1}^{m} [\log x^i + \sum_{j=1}^{|\mathcal{G}|} \sigma_g^{j*} \cdot \log(1 - D(G_j(z^i)))]$
13: Update the weights of discriminator:
$\nabla_{W_D} \frac{1}{m} \sum_{i>m}^{2m} [\log x^i + \sum_{j=1}^{|\mathcal{G}|} \sigma_g^{j*} \cdot \log(1 - D(G_j(z^i)))]$
14: **end for**
Procedure: `SampleFromSupernet`(α)
15: Sample top k subnetworks $\bar{\alpha}$ according to α
16: $\Pi = \text{argmin}_{\Pi \in \bar{\alpha}}$ adv_loss()
Procedure: `SequentialFineTune`()
17: **for** $r = 1, 2, \ldots$ **do**
18: **for** $i = 1, \ldots, |\mathcal{G}|$ **do**
19: $\max_{\theta_{g_i}} \mathbb{E}_z \left[\sum_{j=1}^{|\mathcal{D}|} \sigma_d^{j*} \cdot D_j(G_i(z)) \cdot \Theta \right]$
20: **end for**
21: **for** $j = 1, \ldots, |\mathcal{D}|$ **do**
22: $\max_{\theta_d} \mathbb{E}_x[\log D_j(x)] + \sum_{k=1}^{K} \mathbb{E}_z[\log(1 - D_j(G_k(z)))]$
23: **end for**
24: Augment U^t and find mixed-NE (σ_g^*, σ_d^*)
25: **end for**

Algorithm 2 describes specific changes to adapt DONAS for GANs. We initialize the first generator-discriminator pair G, D and finetune the models in `InitializeModels`() (line 1-2). The two-player oracles are shown in `GeneratorOracle`() and `DiscriminatorOracle`() which describe the best response oracles in the DO-framework of a two-player min-max game with networks (α and β) where the weight of each network must be the best response to the other player.

1) Adapting Adversarial-NAS: GAN optimization process in Adversarial-NAS is defined as a two-player min-max game with value function $V(\alpha, \beta)$ where the weight of each network must be the best response [8]. The min-max game is $\min_\alpha \max_\beta V(\alpha, \beta)$ as:

$$V(\alpha,\beta) = \mathbb{E}_{x\sim p_{data}}(x)[\log D(x|\beta, W_D^*(\beta)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z|\alpha, W_G^*(\alpha))|\beta, W_D^*(\beta))], \quad (3)$$

where the two weights $W_G^*(\alpha), W_D^*(\beta)$ for any architecture pair (α, β) can be obtained through another min-max game between

W_G and W_D, i.e., $\min_{W_{G(\alpha)}} \max_{W_{D(\beta)}} V(W_{G(\alpha)}, W_{D(\beta)})$ as:

$$\begin{aligned} &V(W_{G(\alpha)}, W_{D(\beta)}) \\ &= \mathbb{E}_{x\sim p_{data}}(x)[\log D(x|\beta, W_{D(\beta)}] \\ &\quad + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|\alpha, W_{G(\alpha)})|\beta, W_{D(\beta)}))]. \end{aligned} \tag{4}$$

The weights of generator and discriminator w.r.t. the architecture pair (α, β), $\{W^*_{G(\alpha)}, W^*_{D(\beta)}\}$, can be obtained by a single step of adversarial training as vanilla GANs [8], [30]. The optimal architectures or weights in each iteration can be achieved by ascending or descending the corresponding stochastic gradient by updating in the order of architecture followed by the weights.

2) Generator Oracle: We initialize the generator and sample the noise samples (line 5). Then, we search and update the architecture of the generator by descending its stochastic gradient using the probability distribution σ^*_d for the pure strategies of the discriminator (line 6). Finally, we update the weight of the generator by the stochastic gradient descent while considering σ^*_d (line 7).

3) Discriminator Oracle: Similarly, we initialize the discriminator, sample noise samples and real-data examples (line 11). Then, we update the architecture (line 12) and weights (line 13) by stochastic gradient ascent with the probability distribution σ^*_g for the pure strategies of the generator.

4) Sampling Architecture From Supernet: After the search, we derive the architecture from the resulting supernet (line 15- 16). We sample top k networks selecting maximum values of α and select the networks that give the minimum adversarial loss against the other player.

5) Sequential Finetuning: After the two oracles provide the new strategies for the two players (new generator and discriminator), we sequentially finetune them by Nash distributions σ^{0*}_g and σ^{0*}_d that we obtained from solving the meta-game with linear programming (line 17-22). To finetune multiple generators altogether, we update the objective function of the generator in training with $\mathbb{E}_{(z\sim p_z(z))}[D(G_i(z, \theta_{gi}))]$ and the sequential generators with harmonic mean [33]. Hence, the generator update is:

$$\max_{\theta g_i} \mathbb{E}_{z\sim p_z(z)}[D(G_i(z, \theta_{gi})) \cdot \Phi], \tag{5}$$

where Φ is the harmonic mean of the remaining generators in the array: $\Phi = HM((1-D(G_{i-1}(z, \theta_{g_{i-1}})), \ldots, (1-D(G_1(z, \theta_{g_1}))))]$.

This ensures that the current generator focuses on the data samples from the modes ignored by the previous generators. Arithmetic means and geometric means cannot generalize the ignored data samples well in the case of sampled generators where one of the generators is performing much better. The Harmonic Mean makes sure the currently trained generator focuses on the ignored data samples. The details of sequential finetuning with HM are mentioned in Algorithm 3.

The sequential finetuning of K generators by Harmonic Mean is shown in Algorithm 3. For every epoch, each generator is updated according to Eq. (5) (line 3) followed by the discriminator update where the fake data from the generator is the combination of all K generators (line 5). After the training has finished, we augment the meta-game and solve it to obtain the distribution of K generators to be used in the next iteration.

Algorithm 3 Fintuning K Generators

1: **for** iteration 1,2, ... **do**
2: **for** $i = 1, \ldots, K$ **do**
3: Update the generators using Harmonic Mean: $\max_{\theta_{g_i}} \mathbb{E}_{z\sim p_z(z)}[D(G_i(z, \theta_{gi})) \times HM((1 - D(G_{i-1}(z, \theta_{g_{i-1}}))), \ldots, (1 - D(G_1(z, \theta_{g_1}))))]$
4: **end for**
5: Update the discriminator: $\max_{\theta_d} \mathbb{E}_{x\sim P_{data}}[\log D(x, \theta_d)] + \sum_{k=1}^{K} \mathbb{E}_{z-p_z(z)}[log(1 - D(G_k(z, \theta_{g_k}), \theta_d))]$
6: **end for**

However, HM only tells the ignored samples and cannot give a clear picture of which samples are being captured more than the others at each epoch of the finetuning.

Using the meta-game, which only takes linear complexity to solve, gives Nash distribution of the generators, and hence, the update of the objective function can have a better picture to focus on the less-captured data samples. This method is more effective than focusing heavily on the ignored data samples. Algorithm 2: `SequentialFineTune()` describes the finetuning of multiple generators and discriminators with Nash. Using a similar equation as Eq. (5) and $\Theta = \sigma_g^{i-1*}(1 - D_j(G_{i-1}(z, \theta_{g_{i-1}}))) \times \ldots \times \sigma_g^{1*}(1 - D_j(G_1(z, \theta_{g_1})))$, we finetune each generator (line 19) and discriminator (line 22) using Nash distribution. Instead of using HM, we use the Nash distribution from generating the augmented meta-game and compute mixed NE by linear programming (line 24).

C. DONAS for at

We also propose DONAS for AT where we have the two players as the training classifier $\mathcal{P}_1 = \mathcal{C}$ and the adversarial $\mathcal{P}_2 = \mathcal{A}$. We search the classifier in `ClassifierOracle()` against the perturbed datasets from `AttackerOracle()`. Given L_{val} and L_{train} as loss functions on the validation and training sets respectively, the search process of NAS is defined as a bi-level optimization problem:

$$\begin{aligned} &\min_\theta L_{val}(w^*(\theta), \theta) \\ &\text{s.t. } w^*(\theta) = \text{argmin}_w L_{train}(w, \theta). \end{aligned} \tag{6}$$

For NAS in the classification task, the expensive inner optimization in Eq. (6) is normally approximated by one step training [23] as: $\nabla_\theta L_{val}(w^*(\theta), \theta) \approx \nabla_\theta L_{val}(w - \xi \nabla_w L_{train}(w, \theta), \theta)$ where w denotes the current weights and ξ is the learning rate for a step of inner optimization.

1) Adapting AdvRush: Algorithm 4 describes the specific changes to adapt DONAS for AT. To obtain the first pair of classifier θ and perturbed dataset δ, we perform neural architecture search for θ by AdvRush [28] and set $\delta = 0$ in `InitializeModels()`, then store the parameters in the two arrays $\mathcal{C}$ and $\mathcal{A}$ (line 1-2). For each epoch, we search the new classifier and generate a new perturbed dataset with `ClassierOracle()` and `AttackerOracle()` using the meta-strategies σ^*_c and σ^*_p. We refer to the Free adversarial training (Free AT) algorithm [30], with comparable robustness to the traditional 7-step PGD [25] with significantly faster training. We then convert it to the DO-Framework with the two oracles

Algorithm 4 DONAS for AT

Input: Samples X, learning rate γ, the number of hopsteps m

Procedure: `InitializeModels()`

1: $\theta = \texttt{AdvRushSearch}()$
2: $\delta = 0$

Procedure: `AttackerOracle()`

3: Let $\delta = 0$
4: **for** step $s = 1, 2, ..., \frac{N}{m}$ **do**
5: **for** mini-batch $B \subset X$ **do**
6: **for** $i = 1, 2, \ldots, m$ **do**
7: $g_{adv} = \nabla_{x,\theta\sim(C,\sigma_c^*)} l(x+\delta, y, \theta), x, y \in B$
8: Update δ
$\delta \leftarrow \texttt{Clip}(\delta + \epsilon \cdot sign(g_{adv}), -\epsilon, \epsilon)$
9: **end for**
10: **end for**
11: **end for**

Procedure: `ClassifierOracle()`

12: Initialize architecture and weight (w_0, α_0)
13: **for** iteration $i = 1, 2, ...$ **do**
14: Update weights by descending its stochastic gradient: $\nabla_W L_{train}(w_{i-1}, \alpha_{i-1})$
15: Update the architecture, i.e., α:
$$\begin{cases} \nabla_\alpha [L_{val}(w_i, \alpha_{i-1})], & \text{if } t < \phi \\ \nabla_\alpha [L_{val}(w_i, \alpha_{i-1}) + \gamma L_\lambda], & \text{else} \end{cases}$$
16: **end for**

Procedure: `SampleFromSupernet`(θ)

17: Derive θ through discritization steps from DARTS

Procedure: `FineTune()`

18: **for** steps $s = 1, 2, ..., \frac{N}{m}$ **do**
19: **for** mini-batch $B \subset X$ **do**
20: **for** $i = 1, 2, \ldots, m$ **do**
21: $g_\theta \leftarrow \mathbb{E}_{(x,y)\in B}[\nabla_{\theta,\delta\sim(\mathcal{A},\sigma_a^*)} l(x+\delta, y, \theta)]$
22: $\theta \leftarrow \theta - \gamma \cdot g_\theta$
23: **end for**
24: **end for**
25: **end for**

to allow multiple adversarial updates to be made to the same images without multiple backward passes by training the same mini-batch for m times.

2) Attacker Oracle: For each iteration, we calculate the adversarial gradient using $\theta \in \mathcal{C}$ sampled with the probability distribution of the classifier's strategies from the meta-game σ_c^* to update δ (line 7-8).

3) Classifier Oracle: We adapt AdvRush to search for the architecture and update the weights. According to the ablation study by [28], we set γ = 0.01 and AdvRush loss term L_λ is introduced at ϕ = 50 which we set as iteration number for warm-up process. (line 15).

4) Sampling Architecture From Supernet: After the search and update of the classifier's architecture and weights, we derive the final architecture (line 15) by running the discretization procedure of DARTS [23].

5) Finetuning: We adversarially train the classifier against the attacker. We update θ with stochastic gradient descent (line 21-22) with δ sampled from $\mathcal{A}$ with σ_a^*. We consecutively train each mini-batch for m times.

Finally, we augment the meta-game by calculating the cross-entropy loss and solve it to obtain the distribution. DONAS-AT ends when terminating criteria is satisfied meaning both oracles have searched the best response strategies.

V. Experiments

We conduct our experiments on a machine with Xeon(R) CPU E5-2683 v3@2.00GHz and 4× Tesla v100-PCIE-16GB running Ubuntu operating system. We extend to NAS architectures such as Adversarial-NAS, AdvRush and evaluate against SOTA architectures such as AutoGAN, AlphaGAN, ResNet-18 and RobNet. We adopt the parameter settings and criterion of the baselines as published. We set s = 10 unless mentioned otherwise. We compute the mixed NE of the meta-game with Nashpy.

A. Experiments on DONAS for GAN

We first conduct an experiment to evaluate DONAS-GAN on the datasets: CIFAR-10 [20], STL-10 [6], TinyImageNet [22] and CelebA [24] and evaluate the quantitative performance with FID score against the baselines AutoGAN [10], Adversarial NAS [8] and AlphaGAN [31] to compare with our DONAS-GAN variants where we investigated ablation studies with different approaches varying the number of generators and discriminators.

1) Variants of DONAS-GAN: Initially, we reproduce the Adversarial-NAS search which searches 1 generator followed by sampling multiple top networks from the supernet after the search. We run the experiments for DONAS-GAN/Ind., DONAS-GAN/HM and DONAS-GAN/Nash with K, where K is the number of generators, to select a sample of multiple K generators instead of 1 maximum and finetune the generators. We set the number of discriminators as 1. DONAS-GAN/Ind. uses independent finetuning where all K generators are trained independently whereas DONAS-GAN/HM and DONAS-GAN/Nash use sequentially finetuning where the sampled K generators are finetuned by Harmonic Mean (HM) and Nash. Next, we search K generators iteratively in DONAS-GAN/Iter. and sample the architecture with the maximum α instead of a one-time search. To avoid having the searching architecture for the same batches, we shuffle the train data for each iteration. After the iterative process, we finetune the resulting K generators by Nash. Finally, we perform experiments with the full-version of DONAS-GAN with the pruning limit of K for both generators and discriminators. We set K to prune the generator and discriminator arrays $\mathcal{G}$ and $\mathcal{D}$, and the terminating condition ϵ is now set as $5e^{-3}$.

2) Experiments Results: Table II shows the experiment results with various methods and datasets investigated on DONAS-GAN. We initially set up the experiment for the CIFAR-10 dataset and recorded the FID scores: AutoGAN as 12.42, ALphaGAN as 10.35, Adversarial-NAS as 16.35 for the searched architecture and 10.87 after finetuning. For the variants of DONAS-GAN, the results indicate that our variants bring significant improvements over AdversarialNAS to the results. While DONAS-GAN Iter. suffers from long finetuning time without terminating condition recording 60.88

TABLE II
GENERATIVE RESULTS FOR DONAS-GAN

Methods			AutoGAN	AdvNAS	AlphaGAN	DONAS-GAN				
						Ind.	HM	Nash.	Iter.	Full
CIFAR-10	Search space		10^5	10^{38}	2×10^{11}	10^{38}				
	Search	5	-	16.35	-	14.22	14.22	14.22	14.43	-
		10				14.04	14.04	14.04	13.89	-
	Finetune	5	12.42	10.87	<u>10.35</u>	10.62	9.27	9.06	9.12	**8.93**
		10				10.33	9.32	9.19	9.1	**8.93**
STL-10	Search	5	-	32.48	-	29.16	29.16	29.16	30.02	-
		10				28.95	28.95	28.95	28.99	-
	Finetune	5	31.01	26.98	**22.43**	28.88	26.44	26.13	26.15	25.31
		10				28.72	26.6	26.17	25.82	<u>24.75</u>
Tiny ImageNet	Search	5	-	17.99	-	17.53	17.53	17.53	-	-
		10				16.91	16.91	16.91	-	-
	Finetune	5	16.21	<u>15.10</u>	-	15.21	13.85	12.36	-	11.18
		10				14.6	13.28	11.94	-	**10.42**
CelebA	Search space			1.1×10^{10}	1.8×10^5	1.1×10^{10}				
	Search	5	-	-	-	3.44	3.43	3	2.94	2.93
		10				3.44	3.45	2.92	2.92	2.93
	Finetune	5	-	3.24	**2.12**	3.11	3.13	2.97	2.91	2.87
		10				2.92	2.97	2.97	2.9	<u>2.86</u>

GPU Hours for $K = 5$ and 109.52 GPU Hours for $K = 10$, DONAS-GAN shows satisfying results terminating within 32.28($K = 5$) and 54.19($K = 10$) GPU Hours for the best run.

Similar promising trends are observed in STL-10 datasets. We also compared DONAS-GAN against baselines for Tiny-ImageNet dataset and recorded 16.21 for AutoGAN, 15.10 for Adversarial-NAS and 11.18($K = 5$), 10.42($K = 10$) for DONAS-GAN showing that DONAS-GAN can capture the diversity of data examples better. We also compared our DONAS-GAN variants following the StyleGAN2 modification (without dynamic selection of intermediate latent space $\mathcal{W}$ for CelebA dataset. Although we did not change the our search space to match AlphaGAN's for fair comparisons, we can see that under the same search space, our DONAS-GAN outperforms AdversarialNAS. Qualitative examples in Appendix I of the Supplementary Material also indicate the realistic image generation and the ability to capture the diversity of the classes without mode-collapse. We also present the runtime comparison of AdversarialNAS, AlphaGAN and our DONAS-GAN on CIFAR-10 dataset in Appendix II of the Supplementary Material.

3) Application to Industrial Use Cases: In addition to traditional datasets with large sample sizes, we demonstrate that our DONAS-GAN is applicable to downstream tasks in industrial settings, such as generating multi-label defect data in less-data scenarios. This capability is particularly useful for augmenting the data samples to train data-driven algorithms and mitigating data imbalance through synthetic sampling.

To validate this, we conducted an experiment using the WaferMap dataset, MixedWM38,[1] which contains only 594 data points of (1×52x52) across 38 classes. Unlike AdversarialNAS, which performs the search process only once, our DONAS algorithm searches for generator models multiple times (K =5,10). This iterative approach ensures that reducing the step size in each oracle does not significantly impact performance. As shown in Figure 2, AdversarialNAS tends to lean to just 'donut' defect samples while our DONAS-GAN effectively generates reliable wafer map samples for multiple types of single defect R3:C1, R1:C3, R3:C3, R4:C3 as well as 2 defects R2:C2 and 3 defects R3:C2, demonstrating its potential for industrial applications.

4) Internal Representation of Networks: To analyze how the representation of the networks evolve and differ from each other, we use the centered kernel alignment (CKA) [19] to measure the similarity of the representation between layers and networks of DONAS-GAN trained on TinyImageNet. In Fig. 1, we visualize the CKA values of layers in the same network of DONAS-GAN as heatmaps. We can see that there is a considerable similarity between the hidden layers as represented by the green area but the similarity goes down at the later layers. The trend indicates that our generation quality progressively improves with depth.

In Fig. 3, we visualize the similarity across the models. The result indicates that the search process of DONAS-GAN can obtain diverse architectures of generators (e.g., Generators 0, 1, and 3) to be used as mixed-architecture mitigating the mode-collapse problem. We present the full network similarity analysis in Appendix III of the Supplementary Material.

B. Experiments on DONAS for AT

Finally, we evaluate the robustness of the architectures searched and adversarially trained by DONAS-AT on CIFAR-10 using FGSM and PGD transparent-box-attacks [25]. In

[1] https://github.com/Junliangwangdhu/WaferMap

TABLE III
ROBUST ACCURACY UNDER FGSM AND PGD ATTACKS

Methods		ResNet-18	RobNet			AdvRush	DONAS-AT
			Large	LargeV2	Free		
CIFAR-10	FGSM	49.81%	54.98%	57.18%	59.22%	60.87%	**62.31%**
	PGD-20	45.86%	49.44%	50.53%	52.57%	53.07%	**59.57%**
	PGD-100	45.10%	49.24%	50.26%	51.14%	52.80%	**57.20%**
SVHN	FGSM	88.73%	93.01%	-	93.04%	94.95%	**95.91%**
	PGD-20	69.51%	86.52%	-	88.50%	91.14%	**91.40%**
	PGD-100	46.08%	78.16%	-	78.11%	90.48%	**91.83%**
Tiny ImageNet	FGSM	16.08%	22.16%	-	23.11%	25.20%	**28.11%**
	PGD-20	13.94%	20.40%	-	21.05%	23.58%	**26.30%**
	PGD-100	13.96%	19.90%	-	20.87%	22.93%	**24.10%**

Note: For ResNet-18 and RobNets, we presented the results from the papers [13], [28].

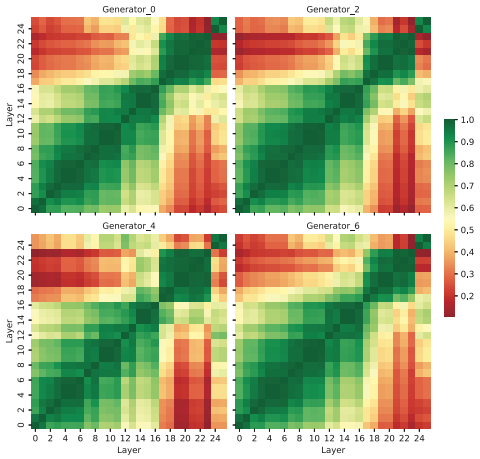


Fig. 1. Linear CKA between layers of the individual searched networks of DONAS-GAN trained on TinyImageNet dataset.

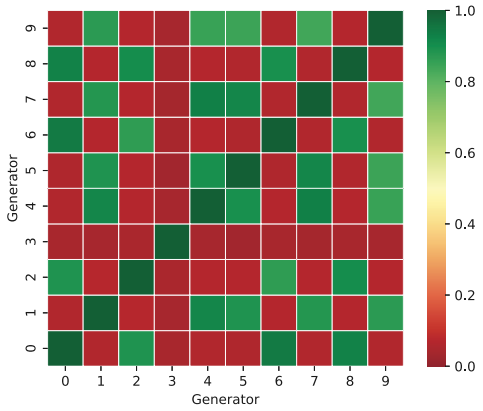
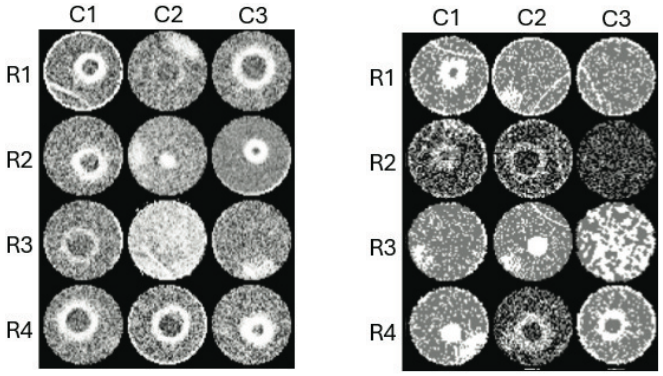


Fig. 3. Linear CKA averaging between the same layers of DONAS-GAN searched networks trained on TinyImageNet. Generators 0, 1 and 3 have different architectures.



(a) AdversarialNAS (b) DONAS-GAN

Fig. 2. Generated wafer images by AdversarialNAS and DONby DONAS-GAN.

particular, we evaluate our proposed DONAS-AT with ResNet-18 [14], variations of RobNet [13] and finally, AdvRush [28] on CIFAR-10, TinyImageNet and SVHN datasets. According to [13], the model's robustness is better as we increase the computational budget and thus, we compare with RobNet-Large and RobNet-LargeV2 which have comparable network parameters (number of channels and cells) to WideResNet. Moreover, we also select RobNet-Free which relaxes the cell-based constraint. After searching the classifier and training in DONAS-AT, we attack the model using PGD attacks where perturbation data is added to input for T iterations (PGD^T). We then use this perturbed input to the model for classification and calculate cross-entropy loss against the target label as criterion. We test the classifiers using PGD^{20} and increase the iterations to 100, i.e., PGD^{100}, to introduce stronger attacks.

Evaluation results are shown in Table III. We observe promising results of DONAS-AT with 62.31% accuracy for FGSM attacks on the CIFAR-10 dataset surpassing the baseline methods such as ResNet-18, RobNet-Large, RobNet-LargeV2, RobNet-Free and AdvRush that record $49.8\%, 54.98\%, 57.18\%, 59.22\%, 60.87\%$. We also evaluated with PGD transparent-box attacks PGD^{20} and PGD^{100}. We obtain 59.57% accuracy on PGD^{20} transparent-box attacks and 57.20% for PGD^{100} observing similar trends which demonstrates the stronger robustness of our approach.

VI. CONCLUSION

In this paper, we propose the double oracle framework solution to NAS for game theoretic deep learning models such as GAN as well as AT. The double oracle framework starts with a restricted game and incrementally adds the best responses

of the player oracles (either for generator/discriminator or classifier/attacker) as the players' strategies. We then compute the mixed NE to get the players' meta-strategies by using a linear program. We leverage two NAS algorithms Adversarial NAS and AdvRush to DO-framework, and introduce sequential finetuning: Harmonic Mean and Nash; to allow the finetuning of multi-models. Extensive experiments with real-world datasets with abundant data such as MNIST, CIFAR-10, SVHN, TinyImageNet and CelebA as well as real-world industry datasets with less data such as MixedWM38, show that our variants of DONAS-GAN and DONAS-AT have significant improvements in comparison to their respective base architectures in terms of both subjective image quality and quantitative metrics.

ACKNOWLEDGMENT

This study partly appeared on Chapter 6 of Aye Phyu Phyu Aung's thesis dissertation.

REFERENCES

[1] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang, "Generalization and equilibrium in generative adversarial nets (GANs)," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 224–232.

[2] A. P. Phyu Aung, X. Wang, R. Yu, B. An, S. Jayavelu, and X. Li, "DO-GAN: A double Oracle framework for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11265–11274.

[3] B. Bošanský, C. Kiekintveld, V. Lisý, J. Cermák, and M. Pechoucek, "Double-Oracle algorithm for computing an exact Nash equilibrium in zero-sum extensive-form games," in *Proc. AAMAS*, May 2013, pp. 335–342.

[4] B. Bosanský, C. Kiekintveld, V. Lisý, and M. Pechoucek, "Iterative algorithm for solving two-player zero-sum extensive-form games with imperfect information," in *Proc. ECAI*, Jan. 2012, pp. 193–198.

[5] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proc. ICLR*, Jan. 2018, pp. 9256–9290.

[6] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.

[7] F. Farnia and A. Ozdaglar, "GANs may have no Nash equilibria," 2020, *arXiv:2002.09124*.

[8] C. Gao, Y. Chen, S. Liu, Z. Tan, and S. Yan, "AdversarialNAS: Adversarial neural architecture search for GANs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5679–5688.

[9] V. Geraeinejad, S. Sinaei, M. Modarressi, and M. Daneshtalab, "RoCo-NAS: Robust and compact neural architecture search," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.

[10] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "AutoGAN: Neural architecture search for generative adversarial networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3223–3233.

[11] I. J. Goodfellow et al., "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 2672–2680.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[13] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, "When NAS meets robustness: In search of robust architectures against adversarial attacks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 631–640.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[15] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung, "MGAN: Training generative adversarial nets with multiple generators," in *Proc. ICLR*, Feb. 2018, pp. 2459–2482.

[16] Y.-P. Hsieh, C. Liu, and V. Cevher, "Finding mixed Nash equilibria of generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2810–2819.

[17] M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe, "A double Oracle algorithm for zero-sum security games on graphs," in *Proc. AAMAS*, May 2011, pp. 327–334.

[18] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.

[19] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3519–3529.

[20] A. Krizhevsky, V. Nair, and G. Hinton. (2009). *CIFAR-10 (Canadian Institute for Advanced Research)*. [Online]. Available: https://www.cs.toronto.edu/kriz/cifar.html

[21] M. Lanctot et al., "A unified game-theoretic approach to multiagent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4190–4203.

[22] Y. Le and X. Yang, "Tiny ImageNet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[23] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. ICLR*, Jan. 2018, pp. 8558–8570.

[24] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.

[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[26] H. B. McMahan, G. J. Gordon, and A. Blum, "Planning in the presence of cost functions controlled by an adversary," in *Proc. ICML*, 2003, pp. 536–543.

[27] L. Mescheder, S. Nowozin, and A. Geiger, "The numerics of GANs," in *Proc. NeurIPS*, vol. 30, Dec. 2017, pp. 1823–1833.

[28] J. Mok, B. Na, H. Choe, and S. Yoon, "AdvRush: Searching for adversarially robust neural architectures," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12302–12312.

[29] S. Saeed Changiz Rezaei et al., "Generative adversarial neural architecture search," 2021, *arXiv:2105.09356*.

[30] A. Shafahi et al., "Adversarial training for free!," 2019, *arXiv:1904.12843*.

[31] Y. Tian, L. Shen, L. Shen, G. Su, Z. Li, and W. Liu, "AlphaGAN: Fully differentiable architecture search for generative adversarial networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6752–6766, Oct. 2022.

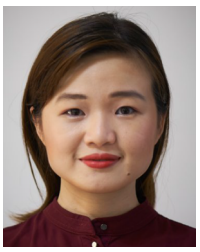
[32] J. Tsai, T. H. Nguyen, and M. Tambe, "Security games for controlling contagion," in *Proc. AAAI*, vol. 26, Sep. 2021, pp. 1464–1470.

[33] S. Varshney, P. K. Srijith, and V. N. Balasubramanian, "STM-GAN: Sequentially trained multiple generators for mitigating mode collapse," in *Proc. Int. Conf. Neural Inf. Process.*, Jan. 2020, pp. 676–684.

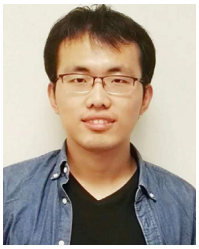
[34] K. Waugh, N. Bard, and M. Bowling, "Strategy grafting in extensive games," in *Proc. NeurIPS*, vol. 22, Dec. 2009, pp. 2026–2034.

[35] T. Xie et al., "Point-NAS: A novel neural architecture search framework for point cloud analysis," *IEEE Trans. Image Process.*, vol. 32, pp. 6526–6542, 2023.

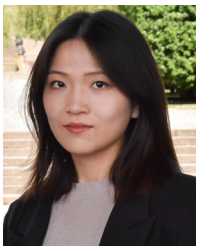
[36] X. Zheng et al., "MIGO-NAS: Towards fast and generalizable neural architecture search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2936–2952, Sep. 2021.



Aye Phyu Phyu Aung received the Ph.D. degree in computer science from Nanyang Technological University (NTU), Singapore, in 2023. Affiliated with the Institute of Infocomm Research (I2R), A*STAR, Singapore, her collaborative research works aim to bridge academia and real-world tech solutions. Her seminal works published in CVPR, ICAPS, and ICIP. Her research interests include generative models, computational game theory, reinforcement learning, and optimization.



Xinrun Wang received the Ph.D. degree from NTU in 2020, under the supervision of Prof. Bo An. He is currently an Assistant Professor with the School of Computing and Information Systems (SCIS), Singapore Management University (SMU). Prior to joining SMU, he has been a Research Assistant Professor with NTU since 2021. Before that, he was a Research Fellow with NTU. His research interests include game theory, (multi-agent) reinforcement learning, and foundation models for decision-making. His work won the Outstanding Student Paper Award from GameSec 2019.



Ruiyu Wang (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the Division of Robotics, Perception, and Learning, KTH Royal Institute of Technology, under the supervision of Asso. Prof. Florian Pokorny. She is supported by the CloudGripper Project within the Wallenberg AI, Autonomous Systems and Software Program (WASP). Her research interests include imitation learning, robotic manipulation, and model generalization.

Hau Chan received the Ph.D. degree in computer science from Stony Brook University in 2015. He completed three years of Postdoctoral Fellowships, including at the Laboratory for Innovation Science, Harvard University, in 2018. He is currently an Assistant Professor with the School of Computing, University of Nebraska–Lincoln. His research aims to establish mathematical and algorithmic foundations to address societal problems from AI and optimization perspectives. He leverages computational game theory and algorithmic mechanism design when the problems involve analyzing/predicting and inducing agent strategic behavior, respectively, and ML and algorithms when the problems deal with predictions and optimizations with non-strategic agents. His research has been supported by NSF, NIH, and USCYBERCOM. He has received several Best Paper Awards from SDM and AAMAS and Distinguished/Outstanding SPC/PC Member recognitions at IJCAI and WSDM. He has given tutorials and talks on computational game theory and mechanism design at venues, such as AAMAS and IJCAI, since 2018, including an Early Career Spotlight at IJCAI 2022. He has served as the Co-Chair for the Doctoral Consortium, Scholarships, and Diversity and Inclusion Activities at AAMAS and IJCAI.

Bo An (Senior Member, IEEE) received the Ph.D. degree in computer science from The University of Massachusetts, Amherst. He is currently a President's Chair Professor of computer science, the Head of the Division of Artificial Intelligence, and the Director of the Centre of AI-for-X, Nanyang Technological University. His research results have been successfully applied to many domains, including infrastructure security, sustainability, and e-commerce. He has published over 150 refereed papers at AAMAS, IJCAI, AAAI, ICLR, NeurIPS, ICML, AISTATS, ICAPS, KDD, UAI, EC, WWW, JAAMAS, and AIJ. His current research interests include artificial intelligence, multiagent systems, computational game theory, reinforcement learning, and optimization. He was elected to the Board of Directors of IFAAMAS, a Senior Member of AAAI, and a Distinguished Member of ACM. He was a recipient of the 2010 IFAAMAS Victor Lesser Distinguished Dissertation Award, an Operational Excellence Award from the Commander, First Coast Guard District of the United States, the 2012 INFORMS Daniel H. Wagner Prize for Excellence in Operations Research Practice, the 2018 Nanyang Research Award (Young Investigator), and the 2022 Nanyang Research Award. His publications won the Best Innovative Application Paper Award from AAMAS 2012, the Innovative Application Award from IAAI 2016, and the Best Paper Award from DAI 2020. He was invited to give an Early Career Spotlight talk at IJCAI 2017. He led the team HogRider which won the 2017 Microsoft Collaborative AI Challenge. He was named to IEEE Intelligent Systems' "AI's 10 to Watch" list for 2018. He was the PC Co-Chair of AAMAS 2020 and the General Co-Chair of AAMAS 2023. He is on the IJCAI Board of Trustees and will be the Program Chair of IJCAI 2027. He is Editor-in-Chief of IEEE INTELLIGENT SYSTEMS. He is an Associate Editor of AIJ, JAAMAS, ACM TAAS, and ACM TIST, and a member of the Editorial Board of JAIR.

Xiaoli Li (Fellow, IEEE) is currently the Department Head and a Senior Principal Scientist with the Institute for Infocomm Research, A*STAR, Singapore. He is also an Adjunct Full Professor with the School of Computer Science and Engineering, NTU. With a diverse range of research interests, he focuses on cutting-edge areas, such as AI, data mining, machine learning, and bioinformatics. His contributions to these fields are evident through his extensive publication record, boasting over 370 peer-reviewed articles, and the recognition he has received, including over ten best paper awards. He is a fellow of Asia–Pacific Artificial Intelligence Association (AAIA). He has been serving as the Editor-in-Chief for the *World Scientific Annual Review of Artificial Intelligence* and an Associate Editor for prestigious journals, such as IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE and *Knowledge and Information Systems*, as well as the Conference Chair and an Area Chair for leading AI, machine learning, and data science conferences, such as AAAI, IJCAI, ICLR, NeurIPS, KDD, and ICDM. Beyond academia, he possesses extensive industry experience, where he has successfully spearheaded over ten research and development projects in collaboration with major industry players across diverse sectors, such as aerospace, telecom, insurance, and professional service companies. He has been recognized as one of the world's top 2% scientists in the AI domain by Stanford University, and Clarivate's Highly Cited Researcher.

J. Senthilnath (Senior Member, IEEE) received the Ph.D. degree in aerospace engineering from the Indian Institute of Science (IISc), India. He is currently a Senior Scientist with the Institute for Infocomm Research (I^2R), Agency for Science, Technology and Research (A*STAR), Singapore. He has published over 100 refereed papers in leading AI/scientific journals/conferences. His research interests include artificial intelligence, generative models, multi-agent systems, online learning, reinforcement learning, optimization, and AI for science. He has held various roles, including an organizing chair, the co-chair, and a program committee member at leading AI conferences. His work has received five best paper awards. He has been recognized as one of the world's top 2% scientists in AI by Stanford University.