# On the Privacy Risks of Cell-Based NAS Architectures

Hai Huang  
CISPA Helmholtz Center for Information Security  
Zikun Zhang  
CISPA Helmholtz Center for Information Security  
Yun Shen  
NetApp  
Michael Backes  
CISPA Helmholtz Center for Information Security  
Qi Li  
Tsinghua University & Zhongguancun Lab  
Yang Zhang  
CISPA Helmholtz Center for Information Security

## ABSTRACT
Existing studies on neural architecture search (NAS) mainly focus on efficiently and effectively searching for network architectures with better performance. Little progress has been made to systematically understand if the NAS-searched architectures are robust to privacy attacks while abundant work has already shown that human-designed architectures are prone to privacy attacks. In this paper, we fill this gap and systematically measure the privacy risks of NAS architectures. Leveraging the insights from our measurement study, we further explore the cell patterns of cell-based NAS architectures and evaluate how the cell patterns affect the privacy risks of NAS-searched architectures. Through extensive experiments, we shed light on how to design robust NAS architectures against privacy attacks, and also offer a general methodology to understand the hidden correlation between the NAS-searched architectures and other privacy risks.

## CCS CONCEPTS
- Security and privacy  
- Computing methodologies → Machine learning;

## KEYWORDS
neural architecture search; membership inference attacks; cell patterns

## ACM Reference Format:

## 1 INTRODUCTION
Deep neural networks (DNNs) have enjoyed a remarkable boom in recent decades and achieved superior performance in many real-world tasks (e.g., image classification [28, 38, 67], object detection [46, 60], text classification [36, 79], etc.). With significant advances in computing power, DNNs have become more complicated. They are both deeper [49, 61] and wider [49, 50] to further improve model performance (e.g., GPT-1/2/3 [4, 56, 57], DALL-E [59], etc.). Despite their success, manually designing those complex networks in a trial-and-error way remains a tedious task, requiring both architectural engineering skills and domain expertise.

Neural architecture search (NAS) [3, 86] is a natural step towards resolving the above challenge. It aims at automating the search process for the most suitable deep neural network architectures for specific tasks. The core idea of NAS is using a search strategy to select an architecture from a predefined search space, and leverage a performance estimation strategy to guide the search process. Directly searching all eligible architectures is computationally expensive; thus the mainstream cell-based NAS methods treat the whole network architecture as a combination of specific modules to reduce the search space (Figure 1 illustrates a typical cell-based NAS architecture). Previous work has shown that the architectures selected by cell-based NAS techniques can attain comparable performance or even outperform those traditional human-designed architectures on tasks such as image classification [23, 39, 87], object detection [22, 72], etc. Existing NAS research mainly focuses on two directions — identify efficient and effective search strategies to obtain the best architecture candidates [2, 10, 73] and improve the robustness of NAS-searched architectures against adversarial examples [15, 24, 41]. On the other hand, although abundant work has already shown that human-designed architectures are prone to privacy attacks, little progress has been made to systematically understand if the NAS-searched architectures are robust to privacy attacks given their complex network topology [54]. In privacy attacks, an adversary’s goal is to gain knowledge of the target model’s training data, which is not intended to be shared. The most popular attack in this domain is membership inference attack (MIA) [64] which enables an attacker to infer whether a sample is used to train a target model. So far, little attention has been paid on the membership privacy of NAS-searched architectures.

Our Work. In this paper, we focus on the aforementioned membership inference attack against NAS-searched architectures due to its pervasiveness and the privacy impact in the real world. Concretely, we address two research questions in this study — whether NAS-searched architectures are robust to membership inference attacks and how architectural information affects such robustness. Note that the first research question has been discussed by some preliminary work [54] in a limited scenario, i.e., label-only membership inference [13, 43].
Figure 1: Illustration of the cell-based NAS architecture. The normal and reduction cells are repeated multiple times and connected together to constitute the whole network architecture, and the cell structures for normal and reduction cells will be searched in the search process of NAS. In the box on the right, there is a general DARTS [44] search space prototype where the arrows with dashed lines represent possible edges and each selected edge will be assigned with one operation indicated by one colored arrow on the right.

To this end, we first conduct a comprehensive measurement study to systematically evaluate the privacy risks of NAS-searched architectures identified by cell-based NAS algorithms, the most popular and influential NAS methods [70]. We find that NAS-searched architectures are generally more robust against MIAs. For instance, 9 out of 10 NAS-searched architectures are more robust to membership inference attacks than all other 10 human-designed architectures for the (Black-Box, Shadow) attack setting (see Section 3 for more details) on the CIFAR10 dataset [1] in our study. Interestingly, our findings show that the robustness against MIAs varies from architecture to architecture. As we see in Figure 3, it is evident that certain NAS-searched architectures are more vulnerable to MIAs than others. For example, architectures searched by TENAS [10] tend to be more prone to membership inference attacks.

Following the new insights from our measurement study, we move on to understand how different internal structures impact the privacy robustness of NAS-searched architectures. To this end, we introduce a general framework to extract cell patterns related to MIA performance (see Section 4). Concretely, we first evaluate the MIA effectiveness on the well-performed architectures, and then extract the cell patterns for the relatively vulnerable and robust architectures separately under the assistance of a regression model. Here, this regression model is trained with the former evaluation results to give direct MIA performance predictions on cell architectures. Finally, we use the extracted cell patterns to modify the internal structures of the target cell architectures and compare the MIA performance before and after the modifications to estimate the effectiveness of the cell patterns. This framework is generic and can be applied to any other analysis work similar to our objective. Following this framework, we evaluate the MIA performance of 2,678 NAS-searched architectures from the NAS-Bench-301 dataset [66] and show that we can identify certain cell patterns that promote or demote MIA performance.

We further apply our analysis framework to mitigating MIA on NAS-searched architectures. The evaluation results show that our cell patterns can successfully promote or demote the MIA performance on the target NAS architectures in the majority of cases. In addition, our cell patterns, though extracted from NAS-Bench-301 dataset, can be transferred to different datasets, attacks, and search spaces. Finally, we show that our work is complementary to existing MIA defense mechanisms when applying the MIA demotion cell patterns to the target NAS architectures, and can further enhance the effectiveness of existing defense strategies. We hope that our findings will inspire future work on designing more robust NAS architectures against membership leakage.

Contributions. In summary, we make the following contributions.

- We systematically evaluate the privacy risks of NAS-searched architectures. Through extensive experiments on four datasets, we show that NAS-searched architectures are usually more robust than human-designed architectures. Our finding is contrary to the results from Pang et al. [54]. However, the privacy risks of the NAS-searched architectures must be individually evaluated.
- We introduce a general framework to analyze the relationship between the NAS-searched architectures and corresponding privacy attacks. We successfully instantiate it to understand, analyze, and identify the hidden cell patterns that impact MIA performance.
- We explore the correlation between the NAS-searched architectures and the robustness against MIAs and identify certain cell patterns that impact MIA performance. The experimental results show that our cell patterns can successfully demote or promote MIA performance on target NAS architectures in most cases. Furthermore, our cell patterns can transfer to new attack settings and are complementary to existing defense work.

2 PRELIMINARIES

2.1 Cell-based Neural Architecture Search

Overview. The search space of NAS can be extremely large if we directly search for and enumerate every single candidate structure of the whole network. To cope with this challenge, cell-based NAS algorithms treat the whole network architecture as a combination of specific small modules, which is referred to as cells. As such, we only need to search for the structures of the basic cells, which significantly reduces the search space and speeds up the search process. Due to the outstanding performance and high flexibility, cell-based NAS algorithms have dominated the NAS research [70]. Figure 1 illustrates a typical cell-based NAS architecture.

There are usually two types of cells in such architectures, normal cell and reduction cell. The normal cell preserves the dimension of the input, while the reduction cell reduces the spatial dimension of the input. The reduction cells are usually placed at the 1/3 and 2/3 positions of the total number of cells, and the rest are normal cells [44]. Under this setting, a cell-based NAS architecture usually has only two reduction cells and tends to have more normal cells especially when the network is deep. Both normal and reduction cells...
are composed of topological combinations of candidate operations (e.g., separable convolution and skip connection).

The discrete candidate operations can be represented as continuous architectural parameters such that the whole architecture can be differentially optimized regarding both the model weights and architectural parameters in the search process. We can gradually train a super network that contains all possible edges in a cell to search for suitable structures of both normal and reduction cells. When we evaluate the performance of candidate architectures, only a limited number (e.g., 2) of input edges with the highest architectural parameters for each intermediate node in the cell will be retained as real corresponding operations. The model weights will be inherited from the super network by the current candidate architecture to evaluate the performance on a validation dataset. In this way, the search space and computation cost are significantly reduced compared to previous NAS methods.

DARTS. We use DARTS [44], the most typical cell-based NAS algorithm, to demonstrate the general concepts and workflow of the cell-based NAS methods. The right box of Figure 1 illustrates the cell search space of the DARTS algorithm, which is represented by a directed acyclic graph (DAG) containing 7 nodes, i.e., 2 input nodes, 4 intermediate nodes, and 1 output node, and multiple edges. The nodes represent the state of the data, and the edges represent the operations on the data. The operation \( o(i,j) \) between node \( i \) and \( j \) is selected from a predefined operation set containing \( K = 7 \) different operations. For an intermediate node \( j \), it obtains the intermediate data \( x(j) \) by aggregating all of its predecessors, i.e., \( x(j) = \sum_{i<j} o(i,j)x(i) \). The outputs of all intermediate nodes will be concatenated to the output node in the end. In the final cell architecture, each intermediate node is only allowed to have 2 input nodes. Therefore, though there are 14 possible edges to connect intermediate nodes in Figure 1, we can only choose \((4 \times 2 = 8)\) edges from them and fill these edges with the most likely operations. For simplicity, we refer to the search space defined in the DARTS algorithm as DARTS search space. Note that, there are also some other kinds of search spaces, the most representative one among them is NAS-Bench-201 [18], a much simpler and smaller search space. More details can be found in Appendix C of our technical report [31].

2.2 Membership Inference Attack

Membership inference attacks (MIAs) against machine learning models aim to infer whether a target sample \( x \) is used to train a target model \( f_{\text{target}} \). As such, MIA directly leads to a privacy breach, allowing the adversaries to learn sensitive information about the training data. For example, in the real world, \( x \) can be a clinical record or an individual. MIA enables the attackers to infer whether this clinical record or individual has been used to train a model associated with a certain disease. This is evidently a privacy and confidentiality violation.

Consider the most common attack setting where the adversary has black-box access to the target model [64], to launch such an attack, the attackers first train a shadow model \( f_{\text{shadow}} \) using a shadow dataset \( \mathcal{D}_{\text{shadow}} \), which performs the same task as \( f_{\text{target}} \) (e.g., classification). The attackers then query \( f_{\text{shadow}} \) using both \( \mathcal{D}_{\text{train}} \) (member data) and \( \mathcal{D}_{\text{test}} \) (non-member data) and obtain the query responses \( R = R_{\text{member}} \cup R_{\text{non-member}} \). They can build an attack model \( f_{\text{attack}} \) : \( R \rightarrow \{0, 1\} \), where the responses of member data are labeled as 1 and those of non-member are labeled as 0. At the attack time, the attackers query \( f_{\text{target}} \) using the data instance \( x \) and use \( f_{\text{attack}} \) to infer whether \( x \in \mathcal{D}_{\text{train}} \) or not using the response from the target model \( f_{\text{target}} \).

3 PRIVACY MEASUREMENT OF NAS-SEARCHED ARCHITECTURES

3.1 Motivation

Many papers point out that overfitting of the target ML models (i.e., the target model performs much better on its training data than test data) is the main factor contributing to the success of MIAs [47, 62, 64, 80]. Shu et al. [65] show that the cell architectures searched by cell-based NAS tend to be shallow and wide to make the loss value converge stably and fast during the search process, which usually leads to competitive generalization performance though it is not guaranteed to be the best. In other words, the architectures searched by these NAS algorithms usually have low overfitting levels. Therefore, a natural hypothesis is that NAS-searched architectures are more robust against MIAs than traditional human-designed ones. Yet, recent work from Pang et al. [54], using empirical results, demonstrates that the NAS architectures are more vulnerable to MIAs than those human-designed architectures, even though the former have better normal model performance than the latter. Note that the conclusion of Pang et al. [54] is based on the label-only scenario for MIA in a black-box setting. In light of the conflicting results, in this section, we comprehensively evaluate the performance of MIAs on NAS-searched architectures and human-designed architectures in both black-box and white-box settings with different levels of knowledge. Our goal is to eliminate the potential experimental bias introduced in the previous research and compare the MIA performance on both categories in a wider spectrum of attack scenarios.

3.2 Measurement Setting

Datasets. We use 4 diverse benchmark datasets to conduct the measurement experiments, including CIFAR10 [1], CIFAR100 [1], STL10 [14], and CelebA [48]. We refer the readers to Appendix A of our technical report [31] for the details of these datasets.

NAS Algorithms. We use 10 representative NAS algorithms to conduct the experiments: (1) DARTS-V1 [44]; (2) DARTS-V2 [44]; (3) ENAS [55]; (4) GDAS [17]; (5) SETN [16]; (6) Random [5]; (7) TENAS [10]; (8) DrNAS [12]; (9) PC-DARTS [78]; (10) SDARTS [11]. We defer the detailed description of them to Appendix A of our technical report [31].

Manual Architectures. For comparison, we also select 10 representative human-designed architectures to conduct the experiments: (1) ResNet [25]; (2) ResNext [76]; (3) WideResNet [82]; (4) VGG [67]; (5) DenseNet [30]; (6) EfficientNet [69]; (7) RegNet [58]; (8) CSPNet [71]; (9) BIT [37]; (10) DLA [81]. We refer the readers to Appendix A of our technical report [31] for the details of these architectures.

Due to space limitation, we defer all the appendices to our technical report [31].
Data Configuration. To facilitate the fair comparison of different MIA methods, we split the original dataset $\mathcal{D}$ into 4 disjoint parts with the same size, i.e., $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{test}}$, $\mathcal{D}_{\text{target}}^t$, and $\mathcal{D}_{\text{shadow}}$. $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ serve as the training and testing dataset of the target model, while $\mathcal{D}_{\text{target}}^t$ and $\mathcal{D}_{\text{shadow}}$ are utilized as the training and testing dataset of the shadow model. As for the NAS algorithm, we further split $\mathcal{D}_{\text{train}}$ into two disjoint parts with the same size to obtain $\mathcal{D}_{\text{train}}^t$ and $\mathcal{D}_{\text{train}}^s$, where $\mathcal{D}_{\text{train}}^t$ and $\mathcal{D}_{\text{train}}^s$ are used as the training and validation datasets respectively of the NAS algorithm in the search process. After the final NAS architecture is generated, we train it from scratch using $\mathcal{D}_{\text{train}}^t$ and test its performance on $\mathcal{D}_{\text{test}}$. Note that our NAS training/validation data split is in line with the latest research by Oymak et al. [53], which states that the train-validation accuracy gap decreases rapidly when the validation data is mildly large. Take CIFAR10 dataset for example, the size of the training dataset of the target model is 7,500, which is fairly sizeable and matching the recommendation by Oymak et al. [53]. In turn, we are more likely to make our NAS-searched architectures achieve better generalization performance.

Attacker's Knowledge. According to the different knowledge levels the attacker has about the target model, we classify the existing MIAs into 5 types and use them to evaluate the above 10 NAS algorithms and 10 human-designed models. In this way, we can conduct a thorough and objective evaluation of both NAS-searched and human-designed architectures in all known MIA attack scenarios [29, 47],

- (Black-Box, Shadow). In this scenario, the attacker only has black-box access to the target model with a local shadow dataset and does not know the training dataset of the target model.
- (Black-Box, Partial). The attacker has black-box access to the target model and has partial knowledge of the training dataset of the target model.
- (White-Box, Shadow). The attacker has white-box access to the target model with only a local shadow dataset and does not know the training dataset of the target model.
- (White-Box, Partial). The attacker has white-box access to the target model and has partial knowledge of the training dataset of the target model. This is the strongest attack scenario.
- (Label-Only). The attacker only has black-box access to the target model and can only infer information from the output labels of the target model, which is the weakest attack scenario.

Note that the attacker obtains all confidence scores returned by the target model except for the (Label-Only) MIA scenario. We defer the training details of the target models and attack models to Appendix A of our technical report [31].

3.3 Measurement Results

Model Performance. The model performance and overfitting of both NAS-search and human-designed models on 4 benchmark datasets are shown in Table 1. We can observe that the performance of the architectures searched by NAS is comparable to or even better than that of the human-designed architectures. Our model performance results are consistent with the previous research from both the ML community [11, 87] and Pang et al. [54]. The overfitting levels on the CIFAR10 and STL10 datasets are relatively high due to the fact that the test accuracy on these two datasets is much lower than that of the CIFAR10 and CelebA datasets, and the training accuracy on all four datasets can reach round 1.0 for most architectures under the same training settings. We can see that, most architectures comply with the general rule that a higher test accuracy tends to lead to a lower overfitting level.

Moreover, we observe that the NAS-searched architectures usually have lower overfitting levels than the human-designed architectures on all four datasets, which leads us to expect the robustness of the NAS-searched architectures against MIAs should be better than that of the human-designed ones, since previous work [47] have shown that a higher overfitting level of the target model usually leads to a better MIA performance. However, this conjecture conflicts with the observations of Pang et al.’s work [54], i.e., the NAS-searched architectures are more vulnerable to MIAs than the human-designed architectures, which motivates us to further explore the reason behind the conflicts and estimate the real privacy threats of various MIAs on the NAS-searched architectures. In fact, Pang et al. [54] only consider the label-only attack scenario in which the attacker has only access to the output labels of the target model, and their data sampling method for testing the MIA performance in their released code implementation3 is actually biased to some data samples with extremely high confidence scores. To test our conjecture and comprehensively study the MIA threats on NAS-searched architectures with reliable and unbiased experimental results, we further evaluate the MIA performance with the aforementioned 5 attack settings on both the NAS-searched and the human-designed architectures.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
<th>STL10</th>
<th>CelebA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet</td>
<td>0.6587 (0.3413)</td>
<td>0.2993 (0.7005)</td>
<td>0.4948 (0.5052)</td>
<td>0.7438 (0.2562)</td>
</tr>
<tr>
<td>EfficientNet</td>
<td>0.6410 (0.3599)</td>
<td>0.3036 (0.4975)</td>
<td>0.4711 (0.3289)</td>
<td>0.7468 (0.2532)</td>
</tr>
<tr>
<td>WideResNet</td>
<td>0.6431 (0.3569)</td>
<td>0.3028 (0.4971)</td>
<td>0.4662 (0.3380)</td>
<td>0.7516 (0.2484)</td>
</tr>
<tr>
<td>VGG</td>
<td>0.7796 (0.2246)</td>
<td>0.4395 (0.5603)</td>
<td>0.6102 (0.3895)</td>
<td>0.7627 (0.2373)</td>
</tr>
<tr>
<td>DenseNet</td>
<td>0.7509 (0.2491)</td>
<td>0.4128 (0.5871)</td>
<td>0.5815 (0.4385)</td>
<td>0.7506 (0.2494)</td>
</tr>
<tr>
<td>EfficientNet</td>
<td>0.5637 (0.4368)</td>
<td>0.2405 (0.7979)</td>
<td>0.3862 (0.6332)</td>
<td>0.7490 (0.2687)</td>
</tr>
<tr>
<td>RegNet</td>
<td>0.5360 (0.4460)</td>
<td>0.2252 (0.7744)</td>
<td>0.4206 (0.3794)</td>
<td>0.7366 (0.2632)</td>
</tr>
<tr>
<td>CSINet</td>
<td>0.6745 (0.3255)</td>
<td>0.3151 (0.6848)</td>
<td>0.5169 (0.4831)</td>
<td>0.7434 (0.2566)</td>
</tr>
<tr>
<td>RIT</td>
<td>0.6165 (0.3833)</td>
<td>0.2417 (0.7581)</td>
<td>0.4274 (0.5726)</td>
<td>0.7401 (0.2599)</td>
</tr>
<tr>
<td>DLA</td>
<td>0.6245 (0.3755)</td>
<td>0.3049 (0.6484)</td>
<td>0.4517 (0.5483)</td>
<td>0.7411 (0.2589)</td>
</tr>
</tbody>
</table>

### Table 1: Normal test accuracy of NAS-searched and human-designed architectures on four different datasets. The numbers in the parentheses stand for the corresponding overfitting levels.

### Table 1: Normal test accuracy of NAS-searched and human-designed architectures on four different datasets. The numbers in the parentheses stand for the corresponding overfitting levels.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
<th>STL10</th>
<th>CelebA</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARTS-V1</td>
<td>0.7043 (0.0979)</td>
<td>0.4071 (0.5929)</td>
<td>0.5917 (0.0911)</td>
<td>0.7668 (0.0272)</td>
</tr>
<tr>
<td>DARTS-V2</td>
<td>0.7028 (0.0962)</td>
<td>0.3895 (0.6105)</td>
<td>0.6123 (0.3877)</td>
<td>0.7704 (0.0777)</td>
</tr>
<tr>
<td>ENAS</td>
<td>0.6345 (0.0670)</td>
<td>0.3895 (0.6057)</td>
<td>0.3726 (0.5062)</td>
<td>0.7664 (0.2331)</td>
</tr>
<tr>
<td>GDAS</td>
<td>0.6621 (0.0657)</td>
<td>0.4113 (0.5956)</td>
<td>0.6357 (0.3540)</td>
<td>0.7639 (0.2311)</td>
</tr>
<tr>
<td>SETN</td>
<td>0.7108 (0.0765)</td>
<td>0.4107 (0.1991)</td>
<td>0.5086 (0.4012)</td>
<td>0.7622 (0.2278)</td>
</tr>
<tr>
<td>Random</td>
<td>0.8112 (0.1886)</td>
<td>0.4335 (0.3671)</td>
<td>0.6452 (0.3517)</td>
<td>0.7659 (0.2032)</td>
</tr>
<tr>
<td>TAN</td>
<td>0.7937 (0.2036)</td>
<td>0.4335 (0.3647)</td>
<td>0.5745 (0.4266)</td>
<td>0.7539 (0.2405)</td>
</tr>
<tr>
<td>InDarts</td>
<td>0.7768 (0.2232)</td>
<td>0.4224 (0.5774)</td>
<td>0.6025 (0.3975)</td>
<td>0.7625 (0.2013)</td>
</tr>
<tr>
<td>PC-DARTS</td>
<td>0.7647 (0.2353)</td>
<td>0.4273 (0.5726)</td>
<td>0.6040 (0.3960)</td>
<td>0.7612 (0.1726)</td>
</tr>
<tr>
<td>SDARTS</td>
<td>0.7641 (0.2359)</td>
<td>0.4410 (0.5485)</td>
<td>0.6400 (0.3680)</td>
<td>0.7727 (0.0790)</td>
</tr>
</tbody>
</table>

datasets are shown in Figure 2, Figure 3, and Figure 4 respectively. The results of the other two attack settings on these three datasets and all attack settings on the CelebA dataset can be found in Appendix B of our technical report [31].

First, we can observe that the MIAs usually have better performance under the white-box settings than the black-box settings. For instance, on the CIFAR10 dataset, the highest AUC score for MIAs on NAS architectures is around 0.81 under the (White-Box, Shadow) setting, while that under the (Label-Only) is about 0.72, and that under the most constrained (Label-Only) settings is around 0.68. We believe that it is due to the abundant additional information (e.g., model weights, gradients) that the attackers extract from the white-box target model. Consequently, they can further manipulate such information to improve the performance of attacks. For example, the attacker can concatenate the gradients of hidden layers of the target white-box model with the output posteriors to serve as the input features for the attack model, which are more likely to enlarge the difference between members and non-members.

Second, given the same target dataset, the same target model tends to perform similarly in different attack settings. We can see that the architectures which obtain relatively high MIA AUC scores in one attack setting are still very likely to acquire relatively high MIA performance in other attack settings, and the histograms of different attack settings on the same dataset are quite similar. This interesting finding is also reasonable. Even though different attack settings might affect attack effectiveness, the majority of information needed by the attack is still offered by the output of the target model. Therefore, the same architecture tends to share similar robustness against MIAs regardless of the various attack settings.
More importantly, we find that the NAS-searched architectures tend to be more robust against various MIAs than the manual ones, which means the latter faces more serious privacy threats than the former. For example, the red bars are usually lower than the blue bars in Figure 2. This further validates our aforementioned conjecture. However, we notice that our conclusion is contrary to the results from Pang et al. [54]. The root cause of such divergence is due to different sampling strategies used to sample member and non-member data. In our case, the distributions of member and non-member data have the same distribution as the original dataset. Pang et al. [54], however, sample data points with high classification confidence scores from the original dataset per their official implementation. Their sampling strategy cannot guarantee the distributions of sampled member and non-member data have the same distribution as the original dataset.

4 ON EXPLORING PRIVACY-RELATED CELL PATTERNS

The empirical experimental results in Section 3 show that the NAS architectures are usually more robust against MIAs; however, different NAS architectures still pose different levels of robustness against MIAs. Some NAS architectures (e.g., DARTS-V2) even appear to be more vulnerable to MIAs than the human-designed VGG on the STL10 dataset. Therefore, in this section, we aim to understand the relationship between the robustness against MIAs and the NAS architectures. Concretely, we seek to address the following questions: (1) What are the common structures among existing NAS architectures which might impact MIAs? (2) Can we find some existing architectural patterns to decrease the MIA risks of the NAS architectures when building NAS? and (3) If we do, can we maintain the model performance while decreasing the MIA risks?

4.1 Overview

Figure 5 illustrates the overall framework for analyzing and evaluating the correlation between the cell patterns and robustness against MIAs of NAS architectures, which consists of three steps.

- **NAS-searched Architecture Preparation.** We first collect a large number of NAS-searched architectures with good model utility and evaluate the corresponding MIA performance on them for subsequent analysis.
- **Cell Pattern Extraction.** We then look deep into the internal cell structures of these NAS-searched architectures and propose a new method to extract common cell patterns that can promote or demote the MIA performance.
- **Cell Architecture Modification.** Finally, we use the extracted cell patterns to modify the internal cell structure of the target architecture to promote or demote the MIA performance on it.

4.2 NAS-searched Architectures Preparation

It is infeasible for us to train thousands of NAS-searched architectures from scratch. Instead, we use NAS-Bench-301 [66], a large scale open-sourced benchmark dataset that contains 59,328 full-trained NAS-searched architectures identified by 17 representative NAS algorithms in a huge DARTS search space (i.e., $10^{18}$ possible architectures) on the CIFAR10 dataset. We drop the redundant architectures which appear multiple times in NAS-Bench-301 and obtain a collection of NAS-searched architectures consisting of 53,558 unique architectures.

As we can see in Table 1, NAS-searched architectures with high test accuracy tend to have low overfitting levels. Their cell patterns are more likely to have better robustness against MIAs while maintaining good model performance at the same time. Besides, in the real world, the end users tend to choose architectures with high model performance. As such, we select the top 5% (i.e., 2,678) architectures with the highest test accuracy scores from NAS-Bench-301. We then evaluate the MIA effectiveness under the most powerful attack setting, i.e., (White-Box, Partial) (see Section 3.2 for details) for each sampled architecture to constitute an Architecture-to-MIA dataset containing the well-performed architectures and their corresponding MIA AUC scores. The MIA AUC scores in this new dataset range from 0.7311 to 0.8773. We group the architectures with MIA AUC scores higher than 0.84 and lower than 0.78 into high (denoted as $\mathcal{A}_{\text{high}}$) and low (denoted as $\mathcal{A}_{\text{low}}$) MIA categories. In total, we have 297 $\mathcal{A}_{\text{high}}$ and 303 $\mathcal{A}_{\text{low}}$ NAS-searched architectures for our cell pattern analysis.

4.3 Cell Pattern Extraction

To get a deeper understanding of the cell structures preferred by the robust or vulnerable NAS architectures, we look into the internal cell structures and try to extract some common cell patterns for specific objectives (e.g., demote the performance of MIAs). Here we take the $\mathcal{A}_{\text{high}}$ and $\mathcal{A}_{\text{low}}$ architectures for cell pattern extraction since they are the most vulnerable and robust architectures respectively in our constructed Architecture-to-MIA dataset.

Operation Distributions. We first analyze the distributions of operations in the normal and reduction cells of these two types of architectures. We have an interesting finding that the convolution

---


---

Figure 5: The overview of our framework exploring the correlation between the NAS architectures and MIA performance.
operations seem critical for model performance while the pooling operations are preferred to mitigate MIAs. As shown in Figure 6, the separable convolutions and skip connections occupy the majority of operations in both normal and reduction cells regardless of the architecture type, which means that these operations are critical to model performance on original tasks. However, we find some changes in the distribution of operations when the architecture type moves from $A_{\text{High}}$ to $A_{\text{Low}}$, especially for specific operations in the reduction cells. Furthermore, according to previous observations in [70], the reduction cell has a relatively small impact on the overall model performance which is dominated by normal cells. And both the $A_{\text{High}}$ and $A_{\text{Low}}$ architectures sampled by us have good model performance, so it is expected that the difference between the operation distributions in the normal cells of these two types of architectures is relatively small. Therefore, we can observe that the change of the operation distributions in the reduction cells is more obvious than that in the normal cells. The frequency of the average pooling $3 \times 3$ operation $\text{ap3}$ increases drastically by 175% when the architecture type changes from $A_{\text{High}}$ to $A_{\text{Low}}$. In general, as for the reduction cells, the convolution operations are preferred by the $A_{\text{High}}$ architectures, while the pooling operations (especially the average pooling operation) are favored by the $A_{\text{Low}}$ architectures. When it comes to the normal cells, even though the operation distributions of both the $A_{\text{High}}$ and $A_{\text{Low}}$ architectures are quite similar to retain model performance, we can still observe that the separable convolution $3 \times 3$ operation $\text{s3}$ is particularly favored by the $A_{\text{High}}$ architectures.

**Operation Importance.** The information obtained from merely the distributions of various operations is far from enough to represent the common cell patterns among the well-performed architectures since the NAS architectures might comply with specific topologies to guarantee model performance. Therefore, we go even further to extract specific cell patterns containing both the topology and operation information. To construct a cell pattern with specific edges, we need to determine the importance of each edge. Following similar strategy in [70], we use Operation Importance (OI) to measure the impact of specific operations with specific edges. For a NAS cell architecture $\alpha$, the directed edge $e_{(i,j)}$ starts from node $i$ to node $j$, we assume that the edge $e_{(i,j)}$ is assigned with operation $o_t$, then the OI metric for $o_t$ on edge $e_{(i,j)}$ is computed as follows:

$$\text{OI}(\alpha, e_{(i,j)} := o_t) = \frac{\sum_{k=1}^{N} f(a_k)}{N} - f(\alpha),$$

where $f(\alpha)$ stands for the MIA AUC score of NAS cell architecture $\alpha$ and $N(\alpha, e_{(i,j)} := o_t)$ represents the neighbor cell set of the original cell architecture $\alpha$.

We regard a cell as a neighbor cell of the original cell architecture $\alpha$ with operation $o_t$ on edge $e_{(i,j)}$ when we either replace operation $o_t$ with another operation $o_p$ ($o_p \neq o_t$) or change the input node $i$ of edge $e_{(i,j)}$ to another preceding node $g(q \neq i)$ in the original cell architecture $\alpha$ to obtain the current cell. OI compares the average MIA performance on the neighbor cells with that on the original cell architecture. A positive value indicates that the current operation in the current edge can mitigate the MIA threats, while a negative value means that the current operation can contribute to MIA performance. Therefore, the attacker prefers the operation-assigned edge with a small negative value, while the defender favors an edge with a large positive value. Note that the MIA evaluation on one cell can be time-consuming (usually takes about 2 hours in our experiments). Even if we consider only modifying a single edge in a cell, it may affect multiple neighbor cells. In turn, iteratively evaluating each neighbor cell after re-training from scratch would be computationally prohibitive when there are many original cells for analysis.

**GIN-based Regression Model.** To speed up the computation process and make this method feasible in reality, we use the aforementioned Architecture-to-MIA dataset to train a regression model to directly predict the MIA performance when the cell architecture is given. The core idea is to build a regression model to achieve the speed-up. The input of the regression model is the architecture, and the output data is its corresponding predicted MIA AUC score. Here we randomly divide the Architecture-to-MIA dataset into three parts with 80%, 10%, and 10% as training, validation, and testing datasets, respectively. We test multiple regression methods (i.e., SVR [19], Random Forest [27], LGBboost [35], XGBoost [9], BANANAS [75], and GIN [77]) to select the model which can best fit the relationship between the NAS architectures and MIA AUC scores. The experiments use the same parameter settings for these regression models as the publicly available implementation of NAS-Bench-301.5

The experimental results are show in Table 2. We use two metrics (i.e., $R^2$ and Spearman$\rho$) to estimate the performance of different regression models. $R^2$ (coefficient of determination) measures how well the observed results are replicated by the model according to the model.

### Table 2: Performance of different regression models trained on the constructed dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>$R^2$</th>
<th>Spearman$\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR</td>
<td>0.1097</td>
<td>0.3055</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.1021</td>
<td>0.3153</td>
</tr>
<tr>
<td>LGBboost</td>
<td>0.0714</td>
<td>0.2783</td>
</tr>
<tr>
<td>XGBoost</td>
<td>0.1332</td>
<td>0.3994</td>
</tr>
<tr>
<td>BANANAS</td>
<td>0.0586</td>
<td>0.2005</td>
</tr>
<tr>
<td>GIN</td>
<td>0.0798</td>
<td>0.3751</td>
</tr>
</tbody>
</table>

5https://github.com/automl/nasbench301.git
to the proportion of the outcome variance successfully explained by the model. SpearmanR (Spearman rank-order correlation coefficient) measures the monotonicity of the relationship between the two datasets (i.e., the ground-truth AUC scores and the predicted AUC scores in our experiments). Higher scores of $R^2$ are preferred and the best possible value of it is 1.0, while scores with higher absolute values are favored for SpearmanR whose best score is $+1/−1$. We can observe that the XGBoost model obtains the best $R^2$ score while the GIN model achieves the best SpearmanR score. Additionally, since the architecture of a NAS cell can be represented as a DAG, and GIN model is a powerful variation of Graph Neural Networks (GNNs) which can learn plentiful information from the architectures of graph data and have surpassed many traditional techniques on graph tasks, it is very reasonable to use GIN in this scenario. As a result, we choose the GIN model as the regression model for further experiments and analysis. Note that we use this GIN-based regression model $f$ to replace $f$ in Equation 1. In turn, we can immediately estimate the MIA AUC score after we get one specific cell architecture.

**Cell Pattern Categories.** According to the goals of different cell patterns, we divide the cell patterns into two categories, i.e., MIA promotion and MIA demotion, where the MIA promotion cell patterns aim to improve the performance of MIAs while the MIA demotion cell patterns attempt to mitigate MIA threats. Take the MIA demotion cell patterns for example, we prefer to choose the edges with a large positive value. Additionally, to accumulate the effectiveness of every single edge in the cell pattern, when constructing the cell pattern one by one edge, we ensure that the selected new edge is adjacent to the current cell pattern (i.e., having nodes in common). In this way, all selected edges are connected as a whole graph. Besides, we make sure that the cell patterns comply with the DARTS cell construction rule, e.g., each intermediate node has at most two input edges.

**Extraction Strategy.** The extraction strategy is shown in Algorithm 1. To focus on the operation-signed edges which are more common among sampled architectures, we only consider and compare the OI scores of those edges appearing more than 14 times in the candidate edge set $E$ when we analyze the cell patterns for the normal or reduction cells of $A_{\text{high}}$ or $A_{\text{low}}$ architectures. Our extraction strategy consists of 4 steps:

- **Initialization.** We separately compute the operation importance for the normal and reduction cells in $A_{\text{high}}$ or $A_{\text{low}}$ architectures, and follow Line 1-7 to prepare for the extraction.
- **Edge Constraint Checking.** We check the number of edges in the current cell pattern in Line 8 and will terminate the extraction process in advance using Line 24 if no more new edge is successfully added to the current cell pattern graph.
- **Construction Rule Checking.** We check whether the current edge is adjacent to the current cell pattern graph and also comply with the rule in the DARTS search space using Line 14.
- **Operation Importance Checking.** We check whether the operation importance score of the current edge meets our requirements using Line 15 and will update the current cell pattern and other data recorders using Line 16-22 if this edge is successfully added.

### Algorithm 1: Cell Pattern Extraction

**Input:** Candidate edge set $E$, maximum number of edges in the cell pattern $L$, demotion flag $\delta$

**Output:** Cell pattern graph $G_p$

1. Initialize the in-degree dictionary $D_{\text{in}}$ with 0 for each node;
2. Initialize the existing cell pattern graph $G_p$, edge topology set $E_p$ and node set $V_p$ as empty sets;
3. if $\delta$ then
   4. Sort $E$ in descending order according to OI scores;
5. else
   6. Sort $E$ in ascending order according to OI scores;
7. Add the input node of the first edge $E_1$ in $E$ to $V_p$;
8. while $|G_p| < L$ do
    9. $\epsilon \leftarrow$ False;
10. $N \leftarrow |E|$;
11. for $i \leftarrow 1$ to $N$ do
    12. // Start node $u$, end node $v$, operation $o$
    13. $(u, v, o) \leftarrow E_i$;
    14. $\tau \leftarrow \text{OI}((u, v, o))$;
    15. if $(u \in V_p \lor v \in V_p)$ and $(u, v) \not\in E_p$ and $D_{\text{in}}[v] < 2$ then
        16. if $(\delta \land \tau > 0)$ or $(\text{not } \delta \land \tau < 0)$ then
            17. $G_p \leftarrow G_p \cup (u, v, o)$;
            18. $E_p \leftarrow E_p \cup (u, v)$;
            19. $V_p \leftarrow V_p \cup \{u, v\}$;
            20. $D_{\text{in}}[v] \leftarrow D_{\text{in}}[v] + 1$;
            21. $E \leftarrow E \setminus (u, v, o)$;
            22. $\epsilon \leftarrow$ True;
        23. break
    24. if not $\epsilon$ then

Note that the NAS cell architectures in the NAS-Bench-301 dataset have 4 intermediate nodes and up to 8 edges in a cell can be modified; thus, we thereby set $L = 8$. We set the demotion flag $\delta$ to “True” to search for the MIA demotion cell patterns, and “False” otherwise to search for the MIA promotion cell patterns.

### 4.4 Cell Architecture Modifications

Through the above efforts, we can extract the cell patterns from the Architecture-to-MIA dataset. Based on these cell patterns, we could go further to modify the internal cell structure of the target architecture to promote or demote MIA performance on it.

**Extracted Patterns.** Figure 7 illustrates the extracted patterns, including both MIA demotion and promotion cell patterns extracted from normal cells and reductions, respectively. We have several interesting findings according to the extracted cell patterns.

- First, separable convolution and dilated separable convolution operations are preferred by the normal cells in both the MIA demotion and promotion cell patterns. As we have discussed before, this is because the model performance is mainly determined by the normal cells and some common operations are necessary for maintaining good model performance.
Second, reduction cells prefer average pooling operations when demoting MIAs but favor convolution operations for MIA promotion. This is also consistent with our findings in the operation distributions of different architectures (see Section 4.3).

Third, the edges in the MIA demotion cell pattern on normal cells are mainly connected between intermediate nodes, while those of other cell patterns are mainly connected to the input nodes. We speculate the reason is that MIA demotion cell patterns use this way to impede direct information transmission for normal cells. According to the statistical results in Figure 6 and the extracted MIA promotion cell patterns in Figure 7, convolution operations seem to be beneficial to MIAs but the normal cells have to use them to retain good model performance even in the MIA demotion cell pattern. To weaken the negative impact of convolutions on privacy, the MIA demotion cell patterns prefer to place these operations in some positions not directly connected to the input nodes to avoid the information with high fidelity being leaked through these operations. On the contrary, max-pooling operations seem to be helpful to mitigate MIAs, so they are mainly directly connected between the input nodes and the intermediate nodes to protect the original information. As for the MIA promotion cell patterns, convolution operations are largely connected to the input nodes to facilitate extracting information with high fidelity.

Cell Architecture Modifications. Our architectural modifications to the target architectures work as follows. First, according to our goal (e.g., promoting MIA performance on the target architecture) and the constraints on our modifications, we constitute corresponding cell patterns from Figure 7 to guide modifications on specific types of cells. For example, when we want to achieve MIA demotion goal, and we are limited to perturb only the normal cell architectures and determine the structure in a cell for up to 4 edges, we select 4 edges from the MIA demotion cell pattern on normal cells following the sequence of the corresponding circled numbers (shown in the left side of Figure 7). We then get the required cell pattern as shown in the upper part of Figure 8.

Second, upon obtaining the cell pattern, we compare the internal structure of both the cell pattern and the normal cell of the target architecture. If an edge $e_{(i,j)}$ with the assigned operation $o_i$ in the cell pattern does not exist in the target normal cell, we replace another existing edge $e_{(k,j)}$ in the target normal cell with the same end node of $e_{(i,j)}$. If there already exists the edge $e_{(i,j)}$ but with a different operation $o_{i'}(o_{i'} \neq o_i)$ in the target normal cell, we simply alter its operation $o_{i'}$ to $o_i$ later. The arrows with dashed lines in Figure 8 represent the original edges ready to be changed.

Finally, we make the modifications to the replacement candidate edges, and replace these edges with the corresponding edges in the cell pattern. The filled double arrows with solid lines in Figure 8 stand for the newly added edges as modifications.

5 CELL PATTERN EVALUATION

5.1 Experimental Setup

Cell Pattern Modification Configuration. We evaluate three categories of cell modifications, namely Only-Reduction, Only-Normal and Dual modifications. Only-Reduction modifications are exclusively made to the reduction cell of the target NAS architecture. Similarly, Only-Normal modifications are made only to the normal cell of the target NAS architecture. Dual modifications indicate that the modifications are made to both reduce and normal cells of
the target architecture. For all these modifications, we use the cell patterns identified in Figure 7.

**Modification Budget.** Note that the cell patterns extracted in Figure 7 tend to include as many edges as possible to either demote or promote MIA performance. However, in the real world, we may not be allowed to modify that many edges in a cell since it would significantly limit the space searchable by various NAS algorithms. That is, the more edges determined by the cell patterns, the fewer edges usable by the NAS algorithm, since the overall amount of edges is fixed. For example, we have 8 edges in total for the DARTS cell (see Figure 1). If the cell pattern has 6 edges, a NAS algorithm has only 2 edges to search. We thereby set a modification budget $m$ to limit the number of edges we can select from one cell pattern. In turn, our final cell patterns used in our experiments contain at most the first $m$ edges from the full cell patterns as shown in Figure 7. We use $m = \{3, 4, 5\}$ to evaluate the effectiveness of cell pattern modifications.

**Runtime Configuration.** Unless otherwise mentioned, we randomly sample 10 NAS-searched architecture instances from our Architecture-to-MIA dataset as the target NAS architectures and average their results. We set $m = 4$ and conduct the experiments on the CIFAR10 dataset by default.

### 5.2 Effectiveness of the Cell Patterns

We evaluate the MIA performance under the (White-Box, Partial) setting on the CIFAR10 dataset. The experimental results for both MIA performance and model utility (i.e., the test performance on the testing dataset for the target model) are shown in Figure 9 and Figure 10 respectively.

The first observation is that our cell patterns can successfully demote or promote MIAs as intended in all cases. Take Only-Reduction modification and modification budget $m = 4$ for instance, the average MIA AUC score of the target architectures drops from 0.8141 to 0.7861. Though a small MIA AUC score decrease from the absolute value perspective, however, such drop can reduce the privacy risks since the real-world models usually requires a huge amount of user data to train.

Our second observation is that it is relatively easier to demote MIA AUC scores of NAS-searched architectures than to promote them. Our hypothesis is that the convolution topology difference between $A_{low}$ and $A_{high}$ may contribute to this phenomenon. To this end, we further count the number of the convolution operations with two different topologies — the direct topology where the convolution operations connect to the input nodes and the intermediate topology where the convolution operations connect two intermediate nodes. For instance, in the MIA demotion cell pattern on the reduction cells in Figure 7, the edge marked with the circled number 6 stands for the intermediate topology, while all the other edges belong to the direct topology. The statistical results for convolution topologies are shown in Figure 11. As we can see in Figure 11, the frequency of direct convolution topology is always higher than that of the intermediate convolutions. And also, the normal cells in the $A_{low}$ architectures contain more intermediate convolution topology than those of the $A_{high}$ architectures. Recall that Section 4.4 shows that the MIA promotion cell pattern on both normal and reduction cells contains many direct convolutions, while the MIA demotion cell patterns on both the normal and reduction cells contain few direct convolutions. Adding more direct convolutions (i.e., promotion) to $A_{high}$ and $A_{low}$ would have less impact on MIA performance due to the fact that the frequency of direct convolution topology is always high in both cases. On the other hand, reducing direct convolutions (i.e., demotion) is more
effective in decreasing the MIA AUC scores, hence more evident in $A_{\text{high}}$ and $A_{\text{low}}$.

Our third observation is that cell patterns with more edges tend to have a larger impact on MIA robustness. For example, when we increase the modification budget $m$ from 3 to 5 in Figure 9b, the MIA performance of the target architectures under the Only-Normal MIA promotion modifications ascend from 0.8148 to 0.8217. Note that, even when the cell patterns have merely $m = 3$ edges, the Only-Normal MIA demotion modifications can still effectively decrease the MIA performance from 0.8141 to 0.7940 in Figure 9a.

Besides, in the MIA demotion scenario, the model utility is even improved after the robustness of the target architecture has been strengthened, which is a desired and satisfying result. The reason behind this phenomenon is that the MIA performance is correlated with the overfitting level of the target model [47]. After we apply the MIA demotion cell patterns, the overfitting level of the target model decreases. The reduced overfitting level helps the target model generalize better and promote the model utility at the test time. For instance, after the Only-Reduction MIA demotion modifications with $m = 4$, the MIA performance in Figure 9a drops from 0.8141 to 0.7861, while the corresponding test accuracy in Figure 10a increases from 0.7866 to 0.7959. The overfitting level of the target model drops from 0.2134 to 0.2041 in this case. Note that, Only-Normal modifications usually have a relatively small impact on the model performance of the target architectures. As we have discussed in Section 4.4, the cell patterns on normal cells have tried to mitigate the side effects on model performance and mainly contain convolution operations.

Finally, Dual modifications usually have the largest impact on the model utility and MIA robustness. For instance, when the modification budget limited to $m = 3$ in Figure 9a, Only-Normal and Only-Reduction MIA demotion modifications can demote the MIA AUC score of the target architectures from 0.8141 to 0.7940 and 0.7946 respectively, while Dual modifications can further reduce the MIA AUC score to 0.7801. At the same time, the test accuracy of the architecture in Figure 10a is improved from 0.7866 to 0.7948, 0.7950, and 0.8061 respectively for Only-Normal, Only-Reduction, and Dual modifications. Note that, we also conduct ablation studies on the impact of the number of cells and the number of intermediate nodes in Appendix D of [31] to get a deeper understanding of the cell patterns. Particularly, we find that deeper networks tend to be more robust while wider networks tend to be more vulnerable to MIAs.

5.3 Loss Contour Analysis

To further investigate the impact of MIA demotion or promotion modifications, we leverage loss contour of the target architecture to understand how those architectures behave after the modifications have been made. We sample one architecture already evaluated in Section 5.2, and plot its training loss contour with regarding to model weight parameters changes using the code implementation$^6$ of Li et al.’s work [40]. Both MIA demotion and promotion modifications are Only-Reduction ones, and the experiments are conducted on the CIFAR10 dataset. The loss contours are shown in Figure 12.

$^6$https://github.com/tomgoldstein/loss-landscape

![Figure 12: The contours before and after MIA demotion or promotion modifications of the target architecture.](image)

The circled gray areas $A$ and $B$ in Figure 12a represent the “trustworthy” areas for MIAs in the loss contour of the original target architecture. We notice that the loss contour tends to be sparse in $A$ and $B$. It means that small perturbations (e.g., training randomness) on the model weights do not affect the loss values of the training dataset much, offering “trustworthy” posteriors of training members for the attacker to discriminate those of non-members. In Figure 12b and Figure 12c, we use arrows to demonstrate the changing trends of the loss contours near the trustworthy areas compared to the original loss contour after the corresponding modifications. We observe that the loss contour near the trustworthy areas of the original loss contour tends to be denser after the MIA demotion modifications. In comparison, the trustworthy areas tend to be sparser after the MIA promotion modifications. The potential reason is that, when the loss contour in these areas becomes denser, little perturbations on the model weights may make the loss values change significantly. The consequence is that many data samples are pushed close to the decision boundary of the current model. In this way, the attacker can only get less confident posteriors from the target model, which hinders the performance of MIAs. And both the MIA demotion and promotion modifications affect the robustness against MIAs by changing the shapes of the loss contour of the target model. Further, our cell patterns have little impact on the model performance, so the overall flatness of the loss contour of the target model has not been changed much.

5.4 Transferability

Our cell patterns are extracted based on the MIA evaluation results with the most knowledgeable (White-Box, Partial) attack setting on the sampled architectures from NAS-Bench-301, a NAS architecture dataset searched in the DARTS search space on the CIFAR10 dataset. Here we want to check whether our cell patterns can transfer to other scenarios. **Transferability among Different Attack Settings.** Our cell patterns are extracted based on the MIA evaluation results with the
Figure 13: Transferring MIA performance of the cell patterns on various MIAs. #1, #2, #3, and #4 represent the (Black-Box, Shadow), (Black-Box, Partial), (White-Box, Shadow) and (Label-Only) attack settings respectively.

(White-Box, Partial) setting, here we would like to see whether our cell patterns also work for other attacks with different attack settings. We change the attack setting for MIAs and re-evaluate the target architectures before and after the MIA demotion and promotion modifications, and the results are shown in Figure 13. We could see that our cell patterns are still effective for all these attacks in all cases, which demonstrates the high transferability of our cell patterns.

Transferability among Different Datasets. Our cell patterns are extracted from the architectures searched on the CIFAR10 dataset, here we want to explore whether our cell patterns are also effective on other datasets. Here we apply our cell patterns to 4 architectures using the last four NAS algorithms in Section 3.2 and searched on CIFAR100, STL10 and CelebA datasets respectively, and evaluate the MIA performance and model utility before and after the architectural modifications.

The experimental results on the CIFAR100 and STL10 datasets are shown in Figure 14 and Figure 15. We defer the evaluation results and corresponding analysis on the CelebA dataset to Appendix E of [31]. It is observed that our cell patterns can still achieve the desired MIA demotion or promotion goals in most cases, which means our cell patterns are also transferable to different datasets. Besides, we can observe that the Only-Reduction modifications tend to have the best performance, while the Only-Normal modifications are inferior to the former. The possible reason is that the model performance of cell-based NAS architectures is mainly determined by the normal cells [70], and the architectures searched on different datasets tend to have different normal cell architectures, which hinders the transferring of the cell patterns on normal cells.

Note that, since our cell patterns are extracted from the DARTS search space, we also test the transferring effectiveness of our cell patterns in the NAS-Bench-201 search space in Appendix F of our technical report [31]. And it is observed that our cell patterns can still partially transfer to the NAS-Bench-201 search space.

5.5 Enhancing Existing Defenses

Existing defenses against MIAs mainly focus on improving the robustness of the target model by masking confidence scores [33, 43, 64], regularization [34, 51, 64], or differential privacy [6, 32, 63], etc. In contrast, our MIA demotion cell patterns are performed in the model architecture design phase and are different from previous defense strategies. Naturally, we wonder if our approach can not only improve the model robustness against MIAs by itself but also complement existing defense techniques when using together?

To this end, we further conduct experiments to estimate whether our MIA demotion cell patterns can enhance existing defense strategies. We choose three representative defense methods which could be applied on both black-box and white-box MIA settings, i.e., data augmentation (DA) [34], label smoothing (LS) [68], and differential privacy (DP) [21]. DA increases the number of training samples by exerting slight modifications to the original data samples and can decrease the overfitting level. LS is a regularization method to mitigate the overconfidence of the target model. DP adds well-calibrated perturbations to the training process of the target model.

The experimental results are shown in Figure 16. We can see that, our MIA demotion cell patterns can successfully enhance the performance of these existing defense strategies in all cases in Figure 16a with a slight impact on the model utility in Figure 16b. To further validate whether our cell patterns can enhance existing defenses even when transferring to other attack settings, we conduct the transferability experiments of other attack settings under defense in Appendix G of our technical report [31]. Overall, our
According to previous experimental results and analysis, we can conclude some general and meaningful guidelines for designing and using more robust cell-based architectures against MIA:

1. **Convolution operations** are important for model performance; however, the architectures with relatively high MIA performance tend to have more convolution operations than others. Therefore, we should use moderate number of convolution operations to strike a tradeoff between model utility and robustness.

2. Use less convolutions directly connected to the input nodes.

3. If more convolution operations are indeed necessary, refer to the second guideline. In this case, constructing a deeper network architecture is recommended.

4. Limit the width of the cell, which usually can be easily done by limiting the number of intermediate nodes in a cell.

## 5.6 Guidelines for Improving Robustness

In this paper, we conduct comprehensive measurement experiments for MIA on both NAS-searched and human-designed architectures, and show that NAS-searched architectures tend to be more robust against MIA. Furthermore, to analyze the hidden cell patterns affecting the robustness against MIA, we design a general framework to extract the cell patterns based on the evaluation results on sampled NAS architectures. We use this framework to extract both MIA demotion and promotion cell patterns from existing well-performed NAS architectures. The experimental results show that our cell patterns can successfully demote or promote the MIA performance on the target architectures and can transfer to various scenarios. Additionally, our MIA demotion cell patterns are model-agnostic and complementary to existing defense techniques and can further enhance the performance of the latter. Finally, we offer some guidelines to design more robust NAS architectures against MIA in the future.

## 7 CONCLUSION

In this paper, we conduct comprehensive measurement experiments for MIA on both NAS-searched and human-designed architectures, and show that NAS-searched architectures tend to be more robust against MIA. Furthermore, to analyze the hidden cell patterns affecting the robustness against MIA, we design a general framework to extract the cell patterns based on the evaluation results on sampled NAS architectures. We use this framework to extract both MIA demotion and promotion cell patterns from existing well-performed NAS architectures. The experimental results show that our cell patterns can successfully demote or promote the MIA performance on the target architectures and can transfer to various scenarios. Additionally, our MIA demotion cell patterns are complementary to existing defense techniques and can further enhance the performance of the latter. Finally, we offer some guidelines to design more robust NAS architectures against MIA in the future.

## ACKNOWLEDGMENTS

We thank our shepherd Xi He and all anonymous reviewers for their constructive comments. This work is partially funded by the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-001 4) and supported by NSFC under Grant 62132011.


