A Recipe for Disaster: Neural Architecture Search with Search Space Poisoning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We assess the robustness of a Neural Architecture Search (NAS) algorithm known as Efficient NAS (ENAS) against data agnostic poisoning attacks on the original search space with carefully designed ineffective operations. By evaluating algorithm performance on the CIFAR-10 dataset, we empirically demonstrate how our novel search space poisoning (SSP) approach and multiple-instance poisoning attacks exploit design flaws in the ENAS controller to result in high prediction error rates for child networks. Furthermore, with just two detrimental operations, our one-shot poisoning approach inflates prediction error rates for child networks up to 90% and 99% on the CIFAR-10 and CIFAR-100 datasets respectively. Our results provide insights into the challenges to surmount in using NAS algorithms with parameter sharing for more adversarially robust architecture search.

1 Introduction

In the modern ecosystem, the problem of finding the most optimal deep learning architectures has been a major focus of the machine learning community. With applications ranging from speech recognition [7] to image segmentation [10], deep learning has shown the potential to solve pressing issues in several domains including healthcare [21; 17] and surveillance [14]. However, a major challenge is to find the best architecture design for a given problem. This relies heavily on the researcher's domain knowledge and involves large amounts of trial and error. More recently, neural architecture search (NAS) algorithms have automated this dynamic process of creating and evaluating new architectures [27; 13; 12]. These algorithms continually sample operations from a predefined search space to construct architectures that best optimize a performance metric over time, eventually converging to the best child architectures. This intuitive idea, outlined in Figure 1, greatly reduces human intervention by restricting human bias in architecture engineering to just the selection of the predefined search space [5].

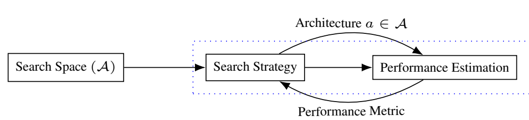

Figure 1: Overview of the NAS framework

Although NAS has the potential to revolutionize architecture search across industry and research applications, human selection of the search space also presents an open security risk that needs to be evaluated before NAS can be deployed in security-critical domains. Due to the heavy reliance of NAS on the search space, poor search space selection either due to human error or by an adversary can potentially impact the training dynamics of NAS severely. This can alter or completely reverse the

Submitted to the 5th Workshop on Meta-Learning at NeurIPS 2021, held virtually. Do not distribute.

predictive performance of even the most optimal final architectures derived from such a procedure. In this paper, we validate these concerns by evaluating the robustness of one such NAS algorithm known as Efficient NAS (ENAS) [16] against data-agnostic search space poisoning (SSP) attacks.

Related Work A comprehensive overview of NAS algorithms can be found in Wistuba et al. [23] and Elsken et al. [5], with Chakraborty et al. [2] summarising advances in adversarial machine learning including poisoning attacks. NAS algorithms have recently been employed in healthcare and applied in various clinical settings for diseases like COVID-19, cancer and cystic fibrosis [20]. Furthermore, architectures derived from NAS procedures have shown state of the art performance, often outperforming manually created networks in semantic segmentation [4], image classification [18; 28] and object detection [28]. With rapid development of emerging NAS methods, recent work by Lindauer and Hutter [11] has brought to light some pressing issues pertaining to the lack of rigorous empirical evaluation of existing approaches. Furthermore, while NAS has been studied to further develop more adversarially robust networks through addition of dense connections [9; 6], little work has been done in the past to assess the adversarial robustness of NAS itself. Search phase analysis has shown that computationally efficient algorithms such as ENAS are worse at truly ranking child networks due to their reliance on weight sharing [26], which can be exploited in an adversarial context. Finally, most traditional poisoning attacks involve injecting mislabeled examples in the training data, which is fairly limited. The expected result is higher prediction error and in some cases a complete reversal of what the network should be predicting. Some examples of traditional poisoning attacks have been executed against feature selection methods [24], support vector machines [1] and neural networks [25]. To the authors' knowledge, no study, has approached poisoning in a data-agnostic manner, especially one that involves poisoning the search space in NAS. In summary, our main contributions through this paper are that:

- We emphasize the conceptual significance of designing adversarial poisoning attacks that leverage the inability of ENAS to alternate between weights shared across effective and ineffective operations through a novel data-agnostic poisoning technique called search space poisoning (SSP) described in Section 3.
- We develop multiple-instance poisoning attacks and design poisoning sets with carefully chosen operations, described in Section 3.2, that cause ENAS to produce child networks with inflated prediction error rates (up to $\sim 85\%$) on image classification tasks.
- We improve upon these results by introducing one-shot poisoning in Section 4, which with just two poisoning operations inflates prediction error rates for child networks up to 90% and 99% on the CIFAR-10 and CIFAR-100 datasets respectively.

2 Background

2.1 Efficient Neural Architecture Search (ENAS)

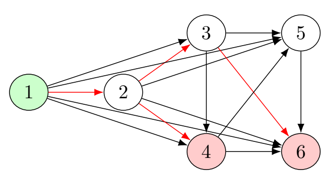

Figure 2: ENAS search space represented as a DAG. Red arrows represent one child model with input node 1 and outputs 4, 6 respectively.

Search Space Consider the set $\mathcal{A}$ containing all possible neural network architectures or child models that can be generated. The ENAS search space is then represented as a directed acyclic graph (DAG) denoted by $\mathcal{G}$ which is a superset of all the child models sampled by ENAS. Every node in Figure 1 represents local computations each having its own parameters with edges representing the flow of information between nodes. Sampled architectures are sub-graphs of $\mathcal{G}$ with parameters being shared amongst child models. The implementation of parameter sharing is the main factor behind ENAS' efficiency as it overcomes the major limitation of NAS. In NAS, all of the architectures were trained from scratch, then upon convergence the trained weights were discarded. So, instead of each child architecture being trained from scratch each sampled model from $\mathcal{G}$ inherits the parameters from previously-trained ones. Throughout this paper, we focus on the highly effective original ENAS search space as outlined in Pham et al. [16] denoted by $\hat{\mathcal{S}}$ = {Identity, 3x3 Separable Convolution, 5x5 Separable Convolution, Max Pooling (3x3), Average Pooling (3x3)}.

Search Strategy The ENAS controller is a predefined long short term memory (LSTM) cell which autoregressively samples decisions through softmax classifiers, where it predicts one hyperparameter at a time, conditioned on previous predictions. The central goal of the controller is to search for optimal architectures by generating a child model $a \in \mathcal{G}$, feeding every decision on the previous step as an input embedding into the next step. Our main search strategy throughout this paper will be macro search where the controller makes two sampling decisions for every layer in the child network: (i) connections to previous nodes for skip connections, and (ii) operations to use from the search space. The model is finally evaluated for its performance which is further used to optimize reward as described next.

Performance Estimation As outlined in Pham et al. [16], ENAS alternates between training the shared parameters ω of the child model $\mathbf{m}$ using stochastic gradient descent (SGD), and parameters θ of the LSTM controller using reinforcement learning (RL). First, keeping ω fixed, θ is trained with REINFORCE [22] and Adam optimiser [8] to maximize the expected reward $\mathbb{E}_{\mathbf{m}\sim\pi(\mathbf{m};\theta)}[\mathcal{R}(\mathbf{m}, \omega)]$ (validation accuracy); and second, keeping the controller's policy $\pi(\mathbf{m}, \theta)$ fixed, ω is updated with SGD to minimize expected cross-entropy loss $\mathbb{E}_{\mathbf{m}\sim\pi}[\mathcal{L}(\mathbf{m}; \omega)]$. It should be noted that different operations associated with the same node in $\mathcal{G}$ have their own unique parameters.

2.2 Training Data Poisoning

Traditionally, training data poisoning is defined as the adversarial contamination of the training set $T \subset \mathcal{D}$ by addition of an extraneous data point $(\mathbf{x}_p, \mathbf{y}_p)$ which maximizes prediction error across training and validation sets, while significantly impacting loss minimization during training [24; 1; 15; 25]. It is assumed here that the data is generated according to an underlying process $f : X \mapsto Y$, given a set $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$ of *i.i.d* samples drawn from $p(X, Y)$, where X and Y are sets containing feature vectors and corresponding target labels respectively. While highly effective, existing poisoning techniques are highly data dependent and operate under the assumption that the attacker has access to training data. A more relaxed assumption would be to decouple the attack modality from training data and make it data agnostic, which is explored in the subsequent section.

3 Search Space Poisoning (SSP)

3.1 General Framework

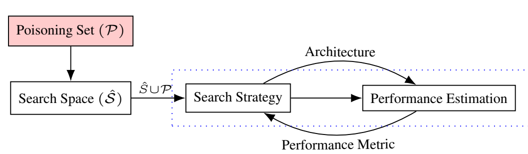

Figure 3: Overview of Search Space Poisoning (SSP)

Motivated by the previously described notion of training data poisoning, we introduce search space poisoning (SSP) focused on contaminating the operation search space. The idea behind SSP is to inject precisely designed ineffective operations into the ENAS search space. Our approach exploits the core functionality of the ENAS controller to sample child networks from a large computational graph of operations by introducing highly ineffective local operations into the search space. On the attacker's behalf, this requires no *a priori* knowledge of the problem domain or training dataset being used, making this new approach more favourable than traditional poisoning attacks. By the same token, SSP could render ENAS prone to human error in search space design. Formally, we describe a poisoned search space as $\mathcal{S} := \hat{\mathcal{S}} \cup \mathcal{P}$, where $\hat{\mathcal{S}}$ denotes the original ENAS search space operations and $\mathcal{P}$ denotes a non-empty set of poisonings where each poisoning is an ineffective operation. An overview of the SSP approach can be observed Figure 3.

3.2 Multiple-Instance Poisoning Attacks

Over the course of training, the LSTM controller paired with the RL search strategy in ENAS develops the ability to sample architectures with operations that most optimally reduce the validation error. We propose multiple-instance poisoning which essentially increases the likelihood of a poisonous operation $o_{\mathcal{P}}$ being sampled from the poisoned search space $\mathcal{S}$. This is achieved by increasing the frequency of sampling $o_{\mathcal{P}}$ from $\mathcal{S}$ through the inclusion of multiple-instances of each $o_{\mathcal{P}}$ from the poisoning multiset, so-called to allow for duplicate elements. An instance factor $q \in \mathbb{N}^{\geq 1}$ represents instance multiplication of $o_{\mathcal{P}}$ in the multiset q times. Henceforth, the probabilities of sampling $o_{\hat{\mathcal{S}}} \in \hat{\mathcal{S}}$ and $o_{\mathcal{P}} \in \mathcal{P}$, respectively, are, $Pr[o_{\hat{\mathcal{S}}}] := \frac{1}{|\mathcal{S}|+q|\mathcal{P}|}$ and $Pr[o_{\mathcal{P}}] := \frac{q}{|\mathcal{S}|+q|\mathcal{P}|}$. From which, it is evident that under a multiple-instance poisoning framework, the probability of sampling poisoned operations is strictly greater than sampling operations in $\hat{\mathcal{S}}$; that is, $Pr[o_{\hat{\mathcal{S}}}] < Pr[o_{\mathcal{P}}]$. It is also important to note that our technique is not the typical image-agnostic perturbation/universal adversarial perturbation $\delta \in \mathbb{R}^d$ intended to fool the target neural network f on almost all the input images from the target distribution X [3]. Finally, another challenge to overcome within our search space poisoning framework is to craft each $o_{\mathcal{P}} \in \mathcal{P}$ such that it counteracts the efficacy of the original operations $o_{\hat{\mathcal{S}}} \in \hat{\mathcal{S}}$, which we tackle in the next section.

3.3 Crafting Poisoning Sets with Operations

Identity Operation The simplest way to attack the functionality of ENAS is to inject non-operations within the original search space which keep the input and outputs intact. As a result, the controller will sample child models with layers representing computations which preserve the inputs, making the operation highly ineffective within a network architecture. This goal is achieved by the inexpensive identity operation which has no numerical effect on the inputs. It should also be noted that, the identity operation is not a skip connection. Therefore, we define our first poisoning set: $\mathcal{P}_1 := \{\text{Identity}\}$.

Gaussian Noise Layer Typically used in signal processing, electronics and mitigating over-fitting as some form of random data augmentation. Gaussian noise is a type of statistical noise which is in the form of a Normal distribution ($X \sim \mathcal{N}(\mu, \sigma^2)$). In PyTorch/Keras, the Gaussian noise layer additively applies zero-centered Gaussian noise passing in the argument of relative standard deviation used to generate the noise. We hypothesize that including such layers with increasingly varied relative standard deviations such as $\sigma = 10$, can significantly impact the accuracy of the generated child models making our poisoning set $\mathcal{P}_2 := \{\text{Gaussian } (\sigma = 10)\}$.

Dropout Layer While dropout layers have historically been shown to be useful in preventing neural networks from over-fitting [19], a high dropout rate can result in severe information loss leading to poor performance of the overall network. This is because given a dropout probability $p \in [0, 1]$, dropout randomly zeroes out some values from the input to decorrelate neurons during training. We hypothesize that including such layers with high dropout probability, such as $p = 1$, has the potential to contaminate the search space with irreversible effects on the training dynamics of ENAS. Here, we define our poisoning set as $\mathcal{P}_3 := \{\text{Dropout } (p = 1)\}$.

Transposed Convolutions As described earlier, amongst other useful operations the original ENAS search space $\hat{\mathcal{S}}$ also contains 3x3 and 5x5 convolutional layers (separable). Intuitively, transposed convolutions upsample the input feature map. It is important to note that transposed convolutions do not perform like deconvolutional layers; they actually swap the forward and backward passes of a convolution. Transposed convolutions, also known as fractionally strided convolutions, stride over the output which is equivalent to a fractional stride over the input. We define our poisoning set: $\mathcal{P}_4 = \{\text{3x3 transposed convolution, 5x5 transposed convolution}\}$.

3.4 Experimental Results

To test the effectiveness of our proposed approach, we designed experiments based on previously described methods outlined in Table 1. Each experiment involved training ENAS on the CIFAR-10 dataset for 300 epochs (hyperparameters used can be found in Appendix A). The results presented

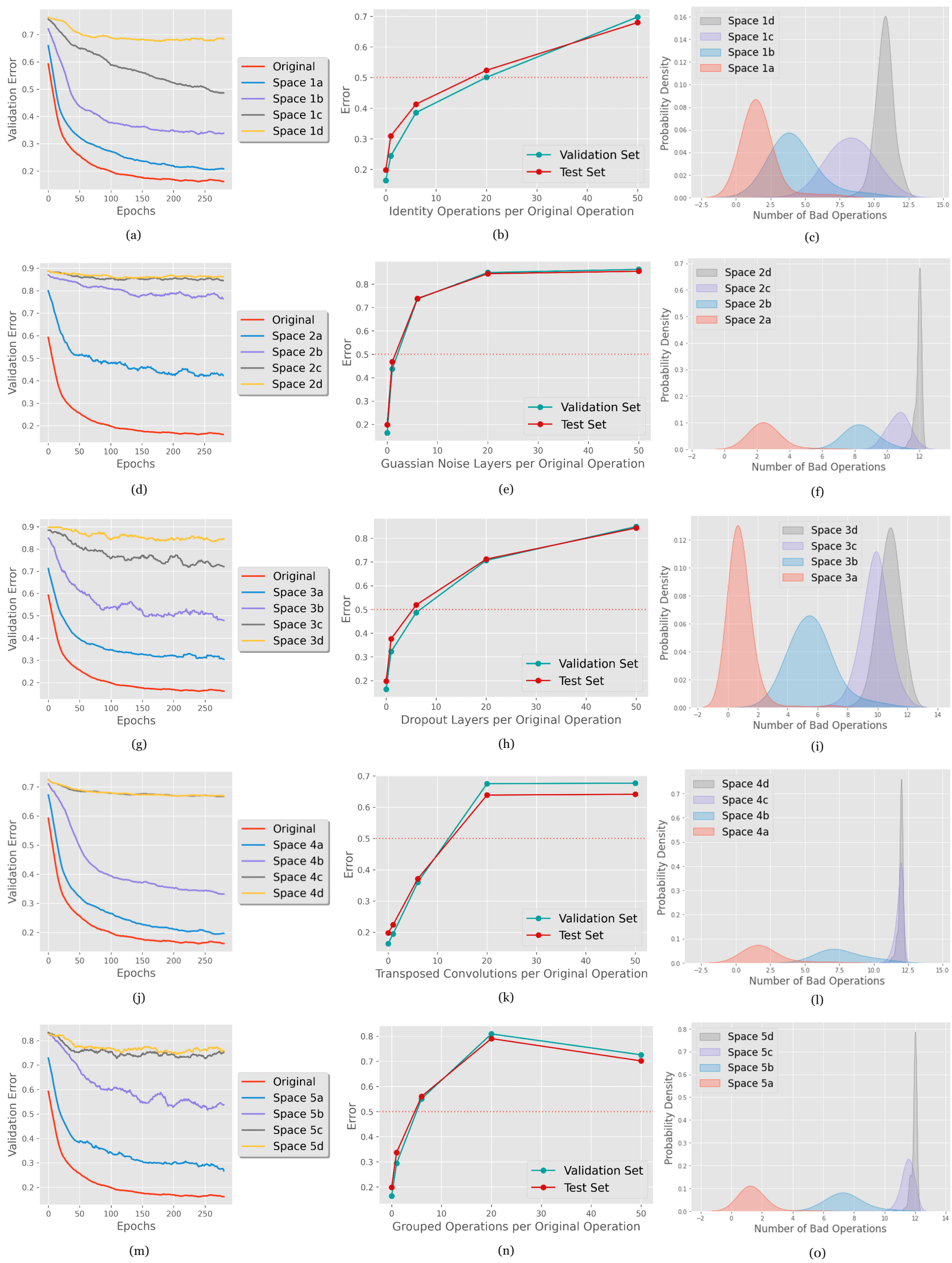

Figure 4: Experimental results for each search space outlined in Table 1. First column represents moving average of the validation error per 20 epochs for 300 total epochs; second column represents final validation and test classification errors as a function of multiple operation instances; and the third column displays the kernel density estimate of the number of bad operations within networks of depth 12 sampled by the ENAS controller over 300 epochs.

Poisoning Set $\mathcal{P}$	Search Space $\mathcal{S}$	Experiment	Poisoning Multiset $q(\mathcal{P})$	Validation Error	Test Error
$\varnothing$	$\hat{\mathcal{S}}$	Original	$\varnothing$	19.53	25.33
		1a	$6(\mathcal{P}_1)$	24.40	30.93
$\mathcal{P}_1 = \{\text{Identity}\}$	$\mathcal{S}_1 = \hat{\mathcal{S}} \cup \mathcal{P}_1$	1b	$36(\mathcal{P}_1)$	38.54	41.31
		1c	$120(\mathcal{P}_1)$	50.07	52.38
		1d	$300(\mathcal{P}_1)$	**69.78**	**67.97**
		2a	$6(\mathcal{P}_2)$	43.71	46.73
$\mathcal{P}_2 = \{\text{Gaussian } (\sigma = 10)\}$	$\mathcal{S}_2 = \hat{\mathcal{S}} \cup \mathcal{P}_2$	2b	$36(\mathcal{P}_2)$	73.64	73.82
		2c	$120(\mathcal{P}_2)$	84.94	84.44
		2d	$300(\mathcal{P}_2)$	**86.26**	**85.49**
		3a	$6(\mathcal{P}_3)$	32.23	37.60
$\mathcal{P}_3 = \{\text{Dropout } (p = 1.0)\}$	$\mathcal{S}_3 = \hat{\mathcal{S}} \cup \mathcal{P}_3$	3b	$36(\mathcal{P}_3)$	48.67	51.88
		3c	$120(\mathcal{P}_3)$	70.63	71.16
		3d	$300(\mathcal{P}_3)$	**84.89**	**84.31**
		4a	$3(\mathcal{P}_4)$	19.50	22.43
$\mathcal{P}_4 = \{$3x3 transposed convolution, 5x5 transposed convolution$\}$	$\mathcal{S}_4 = \hat{\mathcal{S}} \cup \mathcal{P}_4$	4b	$18(\mathcal{P}_4)$	36.00	37.15
		4c	$60(\mathcal{P}_4)$	67.53	63.89
		4d	$150(\mathcal{P}_4)$	**67.68**	**64.14**
		5a	$1(\mathcal{P}_2 + \mathcal{P}_5)$	29.38	33.73
$\mathcal{P}_5 := \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3 \cup \mathcal{P}_4$	$\mathcal{S}_5 = \hat{\mathcal{S}} \cup \mathcal{P}_5$	5b	$6(\mathcal{P}_2 + \mathcal{P}_5)$	55.06	56.02
		5c	$20(\mathcal{P}_2 + \mathcal{P}_5)$	**80.88**	**79.03**
		5d	$50(\mathcal{P}_2 + \mathcal{P}_5)$	72.56	70.14

Table 1: Summary of experimental search spaces with corresponding final validation and test accuracies for SSP. Note that the multiset seed for experiments 5a-5d includes two instances of P_2 to convenient round out the cardinality of the multisets.

in this paper are the average of three runs per experiment. The software used includes Python (3.6.x-3.8.x) and PyTorch (1.9), with CUDA (10.2, 11.1).

Identity Operation Figure 4 shows that multiple-instanced identity operations increase the error considerably. Experiments 1b, 1c, 1d have several identity operations and resulted in high errors, with the extreme 69.19% in experiment 1d. In contrast, experiment 1a only has one identity per original operation and only raised error slightly to 27.28%. These results reinforce our hypothesis laid in the equation in 3.2. Figure 4c illustrates moderate probability distributions of bad operation frequencies across instance-multiplied search spaces.

Gaussian Noise Layer Contaminating the search space with Gaussian noise layers exhibited a similar pattern to identity layers, but the poisoning effect was more dramatic. Experiments with $\sigma = 10$ proved to be quite effective, with the highest instance-factor of 300 producing a final test error of 85.49%. Figure 4f shows that the probability distribution of bad operation frequency is wider at the lower instance factors, but highly concentrated in space 2d around 12 bad operations.

Dropout Layer Instance-multiplying dropout operations exhibits expected behaviour, as seen in Figure 4g. The experiments progressively worsen in error with experiment 3d hitting 83.69% validation and 82.07% test errors. Adding these dropout layers produces similar patterns to previous experiments, with a poisoning effect stronger than that of identity functions, but weaker than that of Gaussian noise layers. This is somewhat unexpected as we had hypothesized that dropout with $p = 1.0$ (discarding all information) would be the most detrimental operation. Here, the distribution of bad operation frequency is lower than previous experiments (Figure 4i).

Transposed Convolutions Adding transposed convolutions has a relatively weaker poisoning effect. Between the first three instance factors ($q = 3, 18, 60$), validation and test errors increase as expected, but between the last two ($q = 60, 150$), it stagnates at a 65.41% test error. Although these errors are quite low in comparison to other experiments, it reinforces the possibility of saturation points, similar to Gaussian noise layers. Figure 4ℓ illustrates a sharp left skew in bad operation frequency distribution just like the Gaussian experiments in 2a-2d in Figure 4f.

Grouped Operations Grouping together our poisoning operations appears to be moderately effective. Over time, the validation error exhibited similar behaviour to our Gaussian noise poisoning spaces using $\mathcal{P}_2$; this is illustrated in Figure 4d and 4m. Reviewing the final errors shown in Figure 4n, we note the high final error of experiment 5c at 79.03% for test. Interestingly, experiment 5d had a much lower test error at 70.14% despite having a higher instance factor $q = 50$, suggesting there exists a point of diminishing returns between $q = 20$ and $q = 50$. Figure 4o shows the distributions of bad operation frequencies having a left skew, similar to those of Gaussian noise and transposed convolution poisoning sets in Figures 4f and 4ℓ.

4 Towards One Shot Poisoning

As seen in Figure 4, the frequency of bad operations far outweighs that of good operations resulting in a left skewed probability distribution for our most effective multiple-instance poisoning experiments. To further improve the attack, we attempt to reduce the number of poisonous operations to a minimum; we call this technique one shot poisoning. In contrast to multiple-instance poisoning, this new approach does not require such high operation frequencies if the poisoning sets are crafted carefully.

In addition to our previous dropout layer, we further infect the original search space $\hat{\mathcal{S}}$ with a widely padded and dilated variant of a convolution from $\hat{\mathcal{S}}$. Our rationale is that dropout operations with $p = 1$ would erase all information and produce catastrophic values such as 0 or not-a-number (`NaN`). The largely dilated kernel in this stretched convolution gives it a wider (but not larger) receptive field to spread these catastrophic values. During backpropagation, the resulting loss and gradient values would similarly be nonsensical causing a ripple effect and leaving the child networks untrained. Since the shared parameters are initially trained from scratch, such child networks would essentially be randomly guessing despite training.

4.1 Experimental Results

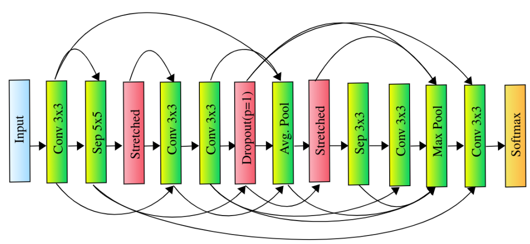

Figure 5: Network produced by ENAS on CIFAR-10 under one shot poisoning. Good and bad operations highlighted in green and red, respectively. The search space used is $\hat{\mathcal{S}} + 2(\mathcal{P}_3^+)$.

To be precise, we define $\mathcal{P}_3^+ := \{\text{Conv}(k = 3, p, d = 50), \text{Dropout}(p = 1)\}$. Our first experiment used the lowest cardinality $|q(\mathcal{P}_3^+)| = 6$ from our previous multiple-instance technique, and our second reduced it to $|q(\mathcal{P}_3^+)| = 2$, the minimum search space with these two bad operations.

Poisoning Set $\mathcal{P}$	Search Space $\mathcal{S} := \hat{\mathcal{S}} + q(\mathcal{P})$	Cardinality $\lvert q(\mathcal{P})\rvert$	Val Error	Test Error
$\varnothing$	$\hat{\mathcal{S}} + \varnothing$	0	19.53%	25.33%
$\mathcal{P}_3 = \{\text{Dropout}(p = 1)\}$	$\hat{\mathcal{S}} + 300(\mathcal{P}_3)$	300	84.89%	84.31%
$\mathcal{P}_3^+ = \{\text{Conv}(k = 3, p, d = 50), \text{Dropout}(p = 1)\}$	$\hat{\mathcal{S}} + 6(\mathcal{P}_3^+)$	6	90.14%	90.00%
	$\hat{\mathcal{S}} + 2(\mathcal{P}_3^+)$	**2**	**90.12%**	**90.00%**

Table 2: Comparing low-instance poisoned search spaces in one shot SSP to our previous multiple-instance poisoning technique shows that one shot SSP can be more effective. Note that in $\mathcal{P}_3^+$ the humble cardinality of 2 roughly matches the 90% test error of cardinality 6.

The results are promising, with error rates shooting up to 90% sharply during training as shown in Table 2 and Figure 6a. Figure 5 shows an example child network produced under this framework. A reasonable frequency of bad operations in child networks can still achieve this 90% test error (Figure 6b). Also, it appears that including stretched convolutions in the poisoning sets tightly constrains the performance of child networks irrespective of instance-factor q. In other words, overwhelming the search spaces with more bad operations has little effect on one shot poisoning. This pattern is consistent in comparing experiments with $\hat{\mathcal{S}} + 6(\mathcal{P}_3^+)$ to $\hat{\mathcal{S}} + 2(\mathcal{P}_3^+)$ in Table 2. This 90% error translates to 10% accuracy, which is roughly equal to randomly guessing, supporting our hypothesis.

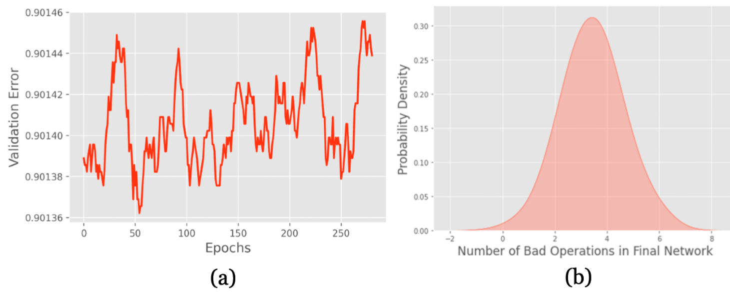

Figure 6: (a) Validation error for one shot poisoning over 300 epochs (b) Distribution of bad operations sampled by the ENAS controller after 300 epochs.

Further Experiments on CIFAR-100 We additionally ran ENAS on the CIFAR-100 dataset to test the efficacy of one shot poisoning. After three runs of 150 epochs, ENAS on the baseline search space resulted in child networks that improve slowly. This is much expected as the task has 100 classes rather than just the 10 from CIFAR-10. Moreover, after running one shot poisoning experiments with the same search space $\hat{\mathcal{S}} + 2(\mathcal{P}_3^+)$, we observe error around 99%, implying an accuracy of 1%. In the context of CIFAR-100, 1% accuracy is roughly equivalent to randomly guessing. A child network produced under this framework is illustrated in Figure 7.

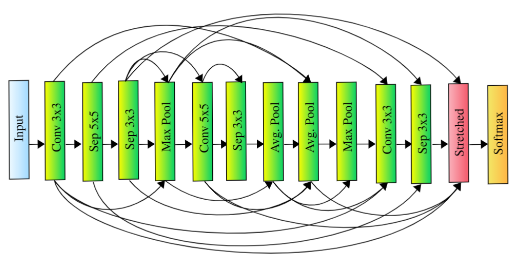

Figure 7: Network produced by ENAS on CIFAR-100 under one shot poisoning. Good and bad operations highlighted in green and red, respectively. The search space used is $\hat{\mathcal{S}} + 2(\mathcal{P}_3^+)$.

The implication is that our one shot poisoning technique causes ENAS to produce child networks that randomly guess. Under one shot poisoning, the training dynamics also exhibit many not-a-number (`NaN`) values for loss and gradient. Given these findings, we suspect that the controller is unable to learn anything from its child networks, and continues to produce randomly-guessing networks.

5 Conclusion

NAS algorithms present an important opportunity for researchers and industry leaders by enabling the automated creation of optimal architectures. However, it is also important to evaluate obvious vulnerabilities in these systems which can result in unforeseen model outcomes if not dealt with beforehand. In this paper, we focused on examining the robustness of ENAS under our newly proposed SSP paradigm. We found that infecting the original search space resulted in child architectures that were highly inaccurate in their predictive abilities. Consistent with the earlier findings in Yu et al. [26], our results highlighted how the controller's dependence on parameter sharing resulted in inaccurate predictions. Moreover, our carefully designed poisoning sets demonstrated the potential to make it easy for an attacker without prior knowledge or access to the training data to still drastically impact the quality of child networks. These findings pave the way for machine learning researchers to explore improvements to the search space and controller design for more adversarially robust search. Finally, our results also present an opportunity for researchers to extend similar ideas to other NAS methods.

References

[1] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[2] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[3] A. Chaubey, N. Agrawal, K. Barnwal, K. K. Guliani, and P. Mehta. Universal adversarial perturbations: A survey, 2020.

[4] L.-C. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. Searching for efficient multi-scale architectures for dense image prediction. *arXiv preprint arXiv:1809.04184*, 2018.

[5] T. Elsken, J. H. Metzen, F. Hutter, et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

[6] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–640, 2020.

[7] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] S. Kotyan and D. V. Vargas. Evolving robust neural architectures to defend from adversarial attacks. *arXiv e-prints*, pages arXiv–1906, 2019.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[11] M. Lindauer and F. Hutter. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, 21(243):1–18, 2020.

[12] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[13] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[14] X. Liu, W. Liu, T. Mei, and H. Ma. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In *European conference on computer vision*, pages 869–884. Springer, 2016.

[15] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38, 2017.

[16] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

[17] F. Piccialli, V. Di Somma, F. Giampaolo, S. Cuomo, and G. Fortino. A survey on deep learning in medicine: Why, how and when? *Information Fusion*, 66:111–137, 2021.

[18] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15 (1):1929–1958, 2014.

[20] M. van der Schaar. Automl and interpretability: Powering the machine learning revolution in healthcare. In *Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference*, pages 1–1, 2020.

[21] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*, 2016.

[22] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[23] M. Wistuba, A. Rawat, and T. Pedapati. A survey on neural architecture search. *arXiv preprint arXiv:1905.01392*, 2019.

[24] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, pages 1689–1698. PMLR, 2015.

[25] C. Yang, Q. Wu, H. Li, and Y. Chen. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340*, 2017.

[26] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019.

[27] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[28] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

Appendix

A. Hyperparameters

HYPERPARAMETER	VALUE
search_for	macro
dataset	CIFAR10 or CIFAR100
n_classes	10 or 100
n_train	45000
n_val	5000
batch_size	128
search_for	300
seed	69
cutout	0
fixed_arc	False
child_num_layers	12
child_out_filters	36
child_grad_bound	5.0
child_l2_reg	0.00025
child_keep_prob	0.9
child_lr_max	0.05
child_lr_min	0.0005
child_lr_T	10
controller_lstm_size	64
controller_lstm_num_layers	1
controller_entropy_weight	0.0001
controller_train_every	1
controller_num_aggregate	20
controller_train_steps	50
controller_lr	0.001
controller_tanh_constant	1.5
controller_op_tanh_reduce	2.5
controller_skip_target	0.4
controller_skip_weight	0.8
controller_bl_dec	0.99
p (Dropout Rate)	1.0

Table 3: Summary of experiment hyperparameters