ET-NAS

Efficient near-Training-free Neural Architecture Search

Domenico Mereu
Politecnico di Torino
Turin, Italy
s302151@studenti.polito.it

Giovanni Mantegna
Politecnico di Torino
Turin, Italy
s296555@studenti.polito.it

Chiara Scagliola
Politecnico di Torino
Turin, Italy
s303037@studenti.polito.it

***Abstract*—We introduce ET-NAS a new algorithm that combines training-free measures and evolutionary search to significantly reduce computational demands and search time. Another contribution is the introduction of the LossSlope, a new near-training-free metric that has a higher correlation with final test accuracy compared to the previously introduced metrics. In our ablation study, we investigate different combinations of training-free measures and we propose a final configuration that provides a good trade-off between fast search and good accuracy. Our code is available here**

I. Introduction

The success of high-performing networks in the last years critically depends on architecture engineering. However, the manual design of new network architecture is extremely consuming in terms of network expertise, time, and resources. Neural Architecture Search (NAS) offers a solution to reduce human efforts in architecture design by automating the process. The automation process proposed in the early NAS approaches was heavily time-consuming and required the training of a large number of architectures. This means that only big research projects involved NAS methods. This search approach recently changed in favour of training-free approaches. Recent research has shown that some measures computed on architecture at his initialization are found to highly correlate with the network's accuracy. These measures can be used as proxy measures for accuracy to avoid the training procedure. This procedure significantly reduces the search cost and makes the NAS accessible for more low resources research.

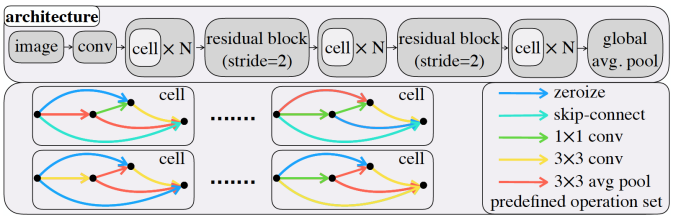
In this work, we present ET-NAS, Efficient near-Training-free Neural Architecture Search, a NAS algorithm with high performance and low computational cost. This new algorithm uses a combination of measures, related to a different aspect of the network, and an evolutionary search algorithm. Our results (Section V-B) show that ET-NAS can be a good trade-off between fast search and good accuracy.

The rest of the paper is organized as follows. In section II we introduce the related works including the main papers used in our research. In section III we make a simple analysis of naswot score and REA algorithm proposed in [1] and [2]. In section IV-A we explain in detail the proxy measures we used in our algorithm. Section IV.B shows the logic of the search algorithm and in section V we make an ablation study and show the final result.

II. Related Works

Neural architecture search (NAS) is a technique for automating the design of artificial neural networks. NAS has been used in the early works to create networks that perform as well as or better than hand-designed architectures by employing a sample-evaluate-update scheme in which a controller generates an architecture and is updated based on the performance of the generated architecture. The ground-breaking research by Zoph et al. (2017) [3] employs an RNN controller to produce descriptions of candidate networks. The quality of the candidates is improved by training networks and using the validation accuracy to update the controller. However, it took them 800 GPUs for 28 days to find the best architecture. Another research by Zoph et al(2018) [4] tried to solve this problem. The idea was to search over neural building blocks, rather than over whole architectures. Mirroring the modular nature of successful hand-designed networks. An alternative approach to NAS is based on evolutionary algorithms. REA [2], instead of reinforcement learning or gradient-based methods, focuses on using evolutionary tournament selection algorithms with an age property that favours younger architectures. In order to reduce the searching time, the research community has recently been focused on the adoption of metrics to grade the various architectures at initialization. A method to assess a neural network without prior training was put forth by Mellor et al. [1] measuring the expressiveness of an architecture focusing on Linear Regions. Moreover, Zero-cost proxies for NAS have just recently been discovered in the pruning-at-initialization literature [5]. These kinds of metrics are designed to score a network architecture at initialization, with the expectation that this score will correlate with the network architecture's final trained accuracy. The scores, which represent the architecture's "saliency," can be either used to replace the costly training stage in conventional NAS or better direct the exploration of current NAS solutions. In order to compare results from different papers the research in the NAS research field has also required the use of some benchmarks like [6]. One of the most used benchmarks is NATS-Bench [7]. This benchmark has two different search spaces, a topology search space and a size search space. The topology search space has a size of 15,625 architectures and each of these architectures is created using five different operations as shown in Fig. 1. All the results obtained are computed on NATS-Bench topology search space

Fig. 1: Example of the architectures skeleton of NATS-Bench



III. Analysis of NASWOT and REA performance

As a preliminary exploratory work, we analyzed the two algorithms proposed in [1] and [2], random search using naswot score and REA. As a first step, we implemented the naswot score as described in the original paper. The key intuition behind the score can be explained as follows. Considering a neural network using rectified linear units as activation functions it is possible to identify a binary indicator that takes into account if a unit is inactive or active. The more similar the binary code associated with two different inputs the more difficult for the network will be to distinguish different data. For the calculation of the score, the authors assuming a mini-batch $\mathbf{X} = \{x_i\}_{i=1}^{N}$ and use this to make a forward operation through the network. Then, they calculate the binary activation $\mathbf{c}_i$ for each data point and the matrix of Hamming distances between each data point in the batch.

$$\mathbf{K}_H = \begin{pmatrix} N_A - d_H(c_1, c_1) & \dots & N_A - d_H(c_1, c_N) \\ \vdots & \ddots & \vdots \\ N_A - d_H(c_N, c_1) & \dots & N_A - d_H(c_N, c_N) \end{pmatrix} \quad (1)$$

Finally, the score is calculated as :

$$s = \log |\mathbf{K}_H| \quad (2)$$

We calculate the naswot score for each network in the topology search space of NATS-Bench [7] using three different datasets: Cifar-10, Cifar-100, ImageNet16-120. According to the results show, we obtain the results in Figure 2(b) which evident the correlation between the score and the test accuracy of the network after the training procedure. Also, the values of Kendall and Spearman correlation are reported in table III of section IV.

A. Naswot score using random search

The authors of [1] use the naswot score on a naive search algorithm. This algorithm is a random search where they consider a set of N architectures, compute the score, and take the architecture with the higher score in the set. We repeat this experiment 30 times for three different values of N. In table I, we reported the average time and standard deviation for a single search and the average and standard deviations test accuracy with 200 epochs.

An interesting comparison can be done running the same algorithm with the validation accuracy at 12 epochs as a score.

TABLE I: Test Accuracy and Time[s] of random search algorithm using naswot score

Dataset	N	Test accuracy	Time
Cifar-10	10	92.59 ± 0.67	1.09 ± 0.19
	100	92.92 ± 0.81	10.81 ± 0.66
	1000	92.99 ± 1.03	108.72 ± 1.42
Cifar-100	10	68.91 ± 2.08	1.67 ± 2.82
	100	69.70 ± 1.40	10.84 ± 0.46
	1000	70.16 ± 0.68	108.79 ± 1.55
ImageNet16-120	10	41.69 ± 3.40	0.78 ± 0.07
	100	42.64 ± 1.26	8.09 ± 1.26
	1000	44.34 ± 2.59	77.6 ± 2.82

Table II has reported the final test accuracy at 200 epochs of the chosen architecture and the time of the overall training procedure. As we can see using validation accuracy at 12 epochs as a score, the algorithm performs better than the naswot score, nevertheless, the time required for the training is about three orders of magnitude higher than the time required to compute naswot score.

TABLE II: Test Accuracy and Time[s] of random search algorithm using 12epochs validation accuracy

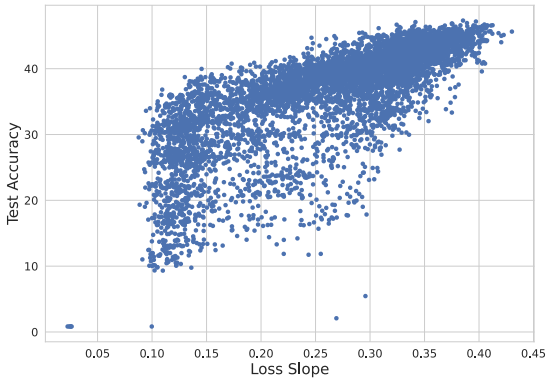
Dataset	N	Test accuracy	Time
Cifar-10	10	92.64 ± 0.94	1134 ± 85
	100	93.98 ± 0.25	11083 ± 191
	1000	94.09 ± 0.28	112872 ± 990
Cifar-100	10	68.94 ± 1.45	2220 ± 227
	100	71.21 ± 1.08	22124 ± 603
	1000	72.11 ± 0.89	220177 ± 1779
ImageNet16-120	10	41.45 ± 2.75	6594 ± 437
	100	44.83 ± 1.50	66670 ± 1740
	1000	46.21 ± 0.58	665714 ± 4956

B. REA algorithm

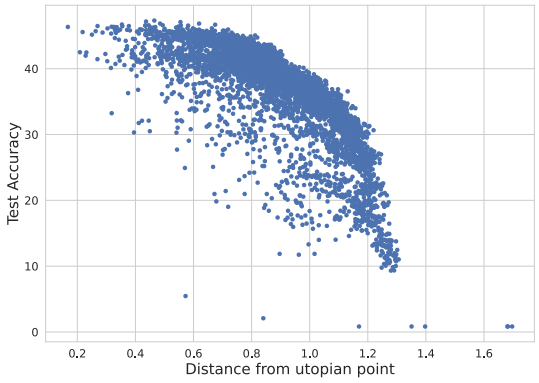
The authors of [1] (Mellor et al.) also use their score in a more complex evolution algorithm. They present a modified version of REA [2], a tournament selection algorithm that uses aging. Their algorithm, AREA is an assisted version of REA in which they use the naswot score in a preliminary phase to reduce the initial poll of architectures. AREA performs slightly better than the original REA. Instead of reproducing AREA, we make two different experiments, we test two version of REA, the first use only the naswot score as a metric to generate child architectures, and the other use 12epochs accuracy. The performances of REA using only naswot to generate child does not perform much better compared to a random search. Instead, using 12epoch accuracy, performances are way better than a random search. We interpret this behavior based on the fact that the naswot score is not enough correlated to test accuracy to direct properly the evolutionary algorithm. For this reason we find a variation of this evolution algorithm using a higher correlated combination of proxy measures.

IV. Proposed Method

We propose a new method called ET-NAS. This new NAS algorithm uses a modified version of REA as a search algo-



(a)



(b)

Fig. 2: Relationship between the values of the scores on the horizontal axis and the test accuracy of ImageNet16-120 on the vertical axis, the scores reported are LossSlope (a), distance from the utopian point (b)

rithm (section IV-B) and some different training free and near training-free measures described in section IV-A.

A. Measures

As anticipated, our main goal is to find a metric capable of scoring a network architecture at initialization, in a way that is reflective of its final trained accuracy. To do so is necessary to consider aspects like trainability and expressivity. To take into account these aspects, a combination of various metrics is necessary. The metrics used in our final algorithm are the following

1) Naswot: As already described in section III, the naswot score was introduced in [1] and considers the expressiveness of a network when it's initialized. The main idea proposed by Mellor et al. is to use rectified linear units (ReLU) as a binary indicator that takes into account whether a unit is active (positive values) or inactive (negative values). The score is calculated as :

$$s = \log |\mathbf{K}_H| \tag{3}$$

where $\mathbf{K}_H$ represents the matrix of Hamming distances between each data point in the batch considered. The correlation between the score and the network's accuracy once trained is shown in Table III.

2) Log Synflow: Synflow was firstly introduced in [5] (Tanaka et al.) to avoid the problem of layer collapse when performing parameter pruning. The idea is to evaluate an architecture by adding the individual weights' contributions. The metric was computed as follows:

$$\mathbf{S}_p(\theta) = \frac{\partial L}{\partial \theta} \odot \theta \tag{4}$$

where L represents the loss function of the network with parameter θ and $\odot$ is the Hadamard product.

Due to the suppression of Batch Normalisation layers in the computation of Synflow, the gradients may be many orders of magnitudes greater than the corresponding weight. To solve this problem a modified version was then introduced in [8] (Cavagnero et al.). To scale down the gradient, a logarithm is applied before summing up all the contributions of each network weight, obtaining the following score function:

$$S(\theta) = |\log \frac{\partial L}{\partial \theta}| \cdot \theta \tag{5}$$

The correlation between the score given by *LogSynflow* and the network's accuracy once trained is shown in Table III.

3) Degree: To better take into account the trainability of a neural network, we also consider the skip connections, essential for the training of very deep networks because they prevent vanishing gradient problem, encountered when training artificial neural networks with gradient-based learning methods and backpropagation. For this reason, we consider as a proxy measure of trainability the sum between the number of convolutional layers and skipped layers, we call this measure the Degree of an architecture.

4) LossSlope: This is a new metric that we propose , the main work related to our metric is GenNAS, initially introduced in [9]. GenNAS adopts a regression-based proxy task using downstream-task-agnostic synthetic signals for network training and evaluation. It was created on the intuition that CNNs produce semantic characteristics in the late layers after learning fine-grained, high-frequency spatial data in the early layers. A Fully Convolutional Network (FCN) is built by eliminating a CNN's final classifier and then extracting the intermediate feature maps for the FCN from various phases. The amount of phases is expressed as N, in our tests N is set to 3. As the output F a feature map tensor is extracted from each stage of the FCN and reshaped through a convolutional

layer. The outputs are the tensors that encapsulate the captured signal patterns from different stages. Synthetic signal tensors are then created for each stage to act as the groundtruth $F*$. Multiple synthetic signal bases are combined to form a synthetic tensor. Good CNNs must be able to learn different frequency signals to capture image features. If a network architecture is good at learning such signals, it is more likely to capture signals from different downstream tasks. To do so, four types of signals were created: 1-D frequency basis (Sin1D), 2-D frequency basis (Sin2D), Spatial basis (Dot and GDot), and Resized input signal (Resize). The MSE loss function for the training is than calculated as follow: $L = \sum_{i=1}^{N} E[(F*_i - F_i)^2]$. During validation, each stage's output importance is weighted and the final score is computed as the last calculated: $L = \sum_{i=1}^{N} 2^{\frac{1}{N-i}} E[(F*_i - F_i)^2]$

In order to reduce calculation time, we propose *LossSlope*. This metric, during the overall training procedure, uses only one batch of data with a batch size of 16 for 10 iterations instead of the 100 proposed in the original version. It speeds up ten times faster the computation. To compute the measure, we fit a straight line to the training loss curve and use the slope as a proxy for learning speed. We changed the sign of the metric to have a positive correlation as shown in Fig 2(a).

In Table III it is possible to see a strong correlation between the score given by *LossSlope* and the network's accuracy after the training.

5) Utopian distance: In our algorithm, we use also a combination of these aforementioned measures. We consider the values of naswot, LogSynflow, and LossSlope all scaled by their max values. We compute the metric for a single architecture considering the distance from the utopian point [1,1,1]. Nearer the architecture to the utopian point higher the test accuracy. Fig 2(b) shows the trend between utopian distance and test accuracy. Also, this metric has a higher correlation compared to the single metrics (Table III)

TABLE III: Kendall and Spearman correlation for the training-free measures

Dataset	Correlation	naswot	LogSynflow	LossSlope	Utopian
Cifar-10	Kendall	0.61	0.63	0.65	**0.74**
	Spearman	0.79	0.83	0.84	**0.91**
Cifar-100	Kendall	0.62	0.61	0.64	**0.74**
	Spearman	0.81	0.81	0.84	**0.90**
ImageNet16-120	Kendall	0.60	0.60	0.62	**0.65**
	Spearman	0.78	0.79	0.82	**0.83**

B. ET-NAS , search algorithm

After the description of the metrics, in this paragraph, we show how we combine all of these to obtain our results. In this section, we describe in detail the structure of the proposed search algorithm. First of all, we generate an initial population of P architectures and score them using the logSynflow. Then from this population, we extract a sample S and select as a parent the architecture with the highest logSynflow. A mutation is applied to the parent architecture to create the child. If the Degree of the child architecture is equal or higher than the Degree of the parent, and if the child architecture is not yet in the network history, then the architecture is added to the newGen pool. The oldest architecture is removed from the population following the aging principle. The evolutionary algorithm runs until the length of $history$ reaches the value of C specified at initialization. Finally, after the evolutionary process, we found a way to select the best architecture in the newGen pool using the Pareto fronts. We compute the naswot score for all the architectures in newGen and using the naswot score and logSynflow we extract the front architectures. At the end of the process, we consider the network closest to the utopian point with coordinates [1,1,1], in the third dimension space of naswot, logSynflow, and LossSlope, (Fig 2(b)) to be the best architecture within the Pareto fronts.

Algorithm 1 ET-NAS

```
population = []
populationSize = P
cycles = C
newGen = []
history = []
for i = 1 : P do
    net = generator.generate()
    logSynflow = net.getLogSynflow()
    population.append((net,logSynflow))
    history.append(net)
end for
while len(history) ≤ cycles do
    Sample population , S , without replacement
    Select the network in S with highest logsynflow as parent
    Mutate parent network to produce child
    degreeChild = child.getDegree()
    degreeParent = parent.getDegree()
    if degreeChild > degreeParent∧child ∉ history then
        Remove oldest network from population
        logSynflow = child.getLogSynflow()
        population.append((child, logSynflow))
        newGen.append((child, logSynflow))
        history.append(child)
    end if
end while
Compute naswot for each network in newGen
Compute ParetoFronts using naswot and logSynflow
Compute SlopeLoss for each network in ParetoFronts
Normalize the three measures in relation to their respective
maximum values.
Take the network closest to the utopian point with coordi-
nates [1,1,1]
```

V. Experiments

A. Ablation Study

In this section we conduct some side experiments on Cifar-10, Cifar-100, and ImageNet16-120, using variations of the main algorithm described in the previous section.

TABLE IV: Comparison of the experiments using variations of the main algorithm evaluated on NATS-Bench. The performances are shown in accuracy with mean±std and time[s], on CIFAR-10, CIFAR-100, and ImageNet-16-120.

	CIFAR-10		CIFAR-100		ImageNet16-120	
Algorithm	Accuracy	Time	Accuracy	Time	Accuracy	Time
RANK	94.24 ± 0.21	29	72.81 ± 0.41	32	**46.48** ± 0.28	22
NO DEGREE	93.98 ± 0.45	14	72.54 ± 0.43	16	45.53 ± 1.07	14
ET-NAS	**94.36** ± 0.02	28	**73.46** ± 0.13	29	46.40 ± 0.19	20

TABLE V: Comparison of several search methods evaluated on NATS-bench. Performances are shown in accuracy with mean±std and time[s], on CIFAR-10, CIFAR-100, and ImageNet-16-120. All the values were performed considering 30 runs on a single RTX 3080 GPU. Our final result is performed on GPU NVIDIA TESLA T4 from Google Colab

	CIFAR-10		CIFAR-100		ImageNet16-120	
Algorithm	Accuracy	Time	Accuracy	Time	Accuracy	Time
REA [2]	94.02 ± 0.31	2e4	72.23 ± 0.84	4e4	45.77 ± 0.80	1.2e5
REINFORCE [3]	93.90 ± 0.26	2e4	71.86 ± 0.89	4e4	45.64 ± 0.78	1.2e5
DARTS [13]	65.38 ± 7.84	2e4	60.49 ± 4.95	4e4	36.79 ± 7.59	1.2e5
ENAS [11]	93.76 ± 0.00	2e4	70.67 ± 0.62	4e4	41.44 ± 0.00	1.2e5
TENAS [12]	93.18 ± 0.39	327	70.24 ± 1.05	327	43.55 ± 8.24	278
TF-MOENAS [10]	93.96 ± 0.10	7753	71.94 ± 0.50	8361	45.68 ± 1.24	9091
GenNAS-N [9]	94.18 ± 0.10	1200	72.56 ± 0.74	1200	45.59 ± 0.54	1200
ET-NAS(ours)	**94.36** ± 0.02	28	**73.46** ± 0.13	29	**46.40** ± 0.19	20

For all the experiments we fix the initial population size P at 20 and the number of cycles C at 60. For the computation of the *LossSlope*, a batch of size 16 is used for 10 iterations, instead, a batch of size 128 is used for naswot. The values of mean and variance of the test accuracy and the average computation time are computed considering 30 runs. The results of the experiments are shown in Table IV. We design our experiments trying to answer the two questions below.

1) **How does the *Degree* influence the score?**

Since our approach relies on choosing architectures with an increasing *Degree* in the evolutionary algorithm, given by the sum between the number of convolutional layers and skipped layers, it is reasonable to question whether our method is dependent on *Degree*. To answer this question we performed our algorithm without the constraint of increasing the *Degree* obtaining the results shown in Table IV under the name *NO DEGREE*, in which it is possible to notice a slight lowering of accuracy but also a more than a halving of the computational time.

2) **What would happen if we considered another way to find the best architecture?**

For this version of the algorithm, the best architecture is the top1 by the sum of three rankings by naswot, *LogSynflow*, and *LossSlope*. Each rank is calculated by sorting architectures in this current subset. The results of this experiment are shown in Table IV under the name *RANK* in which we can see a slight decrease in accuracy for Cifar-10 and Cifar-100 but it achieves a remarkable improvement on ImageNet16-120, achieving 46.48% in accuracy.

B. Results

We conducted 30 replications of each NAS experiment before computing the median and the standard deviation. We consider the same values of population size P and cycles C used in ablation study section. Our approach is able to produce outcomes comparable to or superior to the state-of-the-art performances, as shown in Table V. On Cifar-10, our approach achieves 94.36% accuracy in 28 seconds, outperforming the best training-free metric TF-MOENAS [10], which scored 93.96% accuracy in more than 7500 seconds, and outperforming the best training-based algorithm REA [2] (94.02% accuracy). Our algorithm also outperformed all the others on the Cifar-100 dataset, achieving 73.46% of accuracy, and also on ImageNet16-120, achieving 46.40% of accuracy.

VI. Conculsion

In conclusion, ET-NAS proves to be a good trade-off between accuracy and computational cost. Its near-training-free approach can be used even in situations of constrained time search. The performances that we achieve are above the state-of-the-art NAS methods. In further work, it would be interesting to test the algorithm in a larger benchmark like NAS-Bench-101 [6] in order to test the generalization of our algorithm to other search spaces.

REFERENCES

[1] Mellor, J., Turner, J., Storkey, A., Crowley, E. J. (2021, July). Neural architecture search without training. In International Conference on Machine Learning (pp. 7588-7598). PMLR.

[2] Real, E., Aggarwal, A., Huang, Y., Le, Q. V. (2019, July). Regularized evolution for image classifier architecture search. In Proceedings of the aaai conference on artificial intelligence (Vol. 33, No. 01, pp. 4780-4789).

[3] Zoph, B., Le, Q. V. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.

[4] Zoph, B., Vasudevan, V., Shlens, J., Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8697-8710).

[5] Tanaka, H., Kunin, D., Yamins, D. L., Ganguli, S. (2020). Pruning neural networks without any data by iteratively conserving synaptic flow. Advances in Neural Information Processing Systems, 33, 6377-6389.

[6] Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F. (2019, May). Nas-bench-101: Towards reproducible neural architecture search. In International Conference on Machine Learning (pp. 7105-7114). PMLR.

[7] Dong, X., Liu, L., Musial, K., Gabrys, B. (2021). Nats-bench: Benchmarking nas algorithms for architecture topology and size. IEEE transactions on pattern analysis and machine intelligence.

[8] Cavagnero, N., Robbiano, L., Caputo, B., Averta, G. (2022). FreeREA: Training-Free Evolution-based Architecture Search. arXiv preprint arXiv:2207.05135.

[9] Li, Y., Hao, C., Li, P., Xiong, J., Chen, D. (2021). Generic neural architecture search via regression. Advances in Neural Information Processing Systems, 34, 20476-20490.

[10] Do, T., Luong, N. H. (2021, December). Training-free multi-objective evolutionary neural architecture search via neural tangent kernel and number of linear regions. In International Conference on Neural Information Processing (pp. 335-347). Springer, Cham.

[11] Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In International conference on machine learning (pp. 4095-4104). PMLR

[12] Chen, W., Gong, X., Wang, Z. (2021). Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. arXiv preprint arXiv:2102.11535.

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