OPTIMIZING NEURAL NETWORKS WITH GRADIENT LEXICASE SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Lexicase selection is a parent selection method for which the selection is based on assessing individual cases in random order, instead of using aggregated metrics such as loss and accuracy. While lexicase selection and its variants have demonstrated success in various fields including genetic programming, genetic algorithms, and more recently in symbolic regression and robotics, its applications in other forms of modern machine learning have not yet been explored. In this paper, we aim to investigate the effectiveness of lexicase selection in training deep neural networks and illustrate how its general idea may fit into the context of modern machine learning at scale. We propose Gradient Lexicase Selection, an optimization method that combines stochastic gradient descent and lexicase selection in an evolutionary fashion. Experimental results show that the proposed method improves the generalization performance of popular deep neural network architectures on benchmark image classification datasets, and helps models learn more diverse representations.

1 INTRODUCTION

Modern data-driven learning algorithms generally define an optimization objective, *e.g.*, a fitness function in genetic algorithms (Holland, 1992) or a loss function in deep learning (LeCun et al., 2015), that computes the aggregate performance on the training data for the purpose of acquiring the best parameters in order to solve the task. Taking the image classification problem as an example, most recent approaches use Cross-Entropy loss with gradient descent (Bottou, 2010) and backpropagation (Rumelhart et al., 1985) to train deep neural networks (DNNs). Despite the success that advanced DNNs can reach human-level performance on the image recognition task (Russakovsky et al., 2015), one potential drawback for such aggregated performance measurement is that the model may learn to seek "compromises" during the learning procedure. To give an example, consider a situation that may happen during the training phase for a batch of ten cases: nine cases are correctly predicted with high probabilities, but one is wrong. The aggregated loss may cause the model learning to compromise the wrong case for higher probabilities on other cases, which may lead the optimization process to be stuck at local optima.

This potential optimization problem can be referred to as an uncompromising problem (Helmuth et al., 2014), that is, a problem for which it is not acceptable for a solution to perform sub-optimally on any one of the cases in exchange for better performance on others. It becomes relevant in the context of deep learning where the optimization objective may introduce the behaviors of seeking compromises. In order to improve the generalization of DNNs, it is important to maintain the diversity and generality of the representations contributed by every training case.

From the literature, uncompromising problems have been recently explored in genetic programming (GP) and genetic algorithms (GAs) for tasks such as program synthesis. Among many methods that aim to mitigate this problem, lexicase selection (Helmuth et al., 2014; Spector, 2012) and its variants (Helmuth et al., 2016; Helmuth & Spector, 2015; Helmuth et al., 2014; Liskowski et al., 2015) have been shown to outperform others (Fieldsend & Moraglio, 2015; Galvan-Lopez et al., 2013; Krawiec & Liskowski, 2015) in a number of applications. Instead of using an aggregated fitness function for parent selection, lexicase selection gradually eliminates candidates as it proceeds to look at how the population fares at each data point in a shuffled dataset, in which way it can bolster the diversity and generality in populations. Recent work also shows that lexicase selection can be

Algorithm 1: Lexicase selection to select one parent program in genetic programming

Data:

- `cases` - randomly shuffled sequence of data samples to be used in selection
- `candidates` - the entire population of programs

Result:

- an individual program to be used as a parent

```
while True do
    candidates ← the subset of the current candidates that have exactly best
      performance on the first case in cases
    if candidates contains only one single individual then
        return this individual
    end
    if cases contains only one single test case then
        return a randomly selected indivudual from candidates
    end
    Remove the first case from cases
end
```

used in rule-based learning systems (Aenugu & Spector, 2019), symbolic regression (La Cava et al., 2016), machine learning (La Cava & Moore, 2020b;a), and evolutionary robotics (Huizinga & Clune, 2018; La Cava & Moore, 2018) to improve model generalization, especially in situations of partial or missing data.

In this work, we aim to explore the application of lexicase selection in the task of optimizing deep neural networks. Taking advantage of the commonly-used gradient descent and backpropagation methods, we introduce Gradient Lexicase Selection, an optimization method for training deep neural networks that not only benefit from the efficiency of gradient-based learning but also improves the generalization of the networks using the outline of lexicase selection method in an evolutionary fashion. We test the proposed method on the basic image classification task on three benchmark datasets (CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), and SVHN (Netzer et al., 2011)). Experimental results show that gradient lexicase selection manages to improve the performance of the DNNs consistently across six different popular architectures (VGG (Simonyan & Zisserman, 2015), ResNet (He et al., 2016), DenseNet (Huang et al., 2017), MobileNetV2 (Sandler et al., 2018), SENet (Hu et al., 2018), EfficientNet (Tan & Le, 2019)). In addition, we perform further ablation studies to analyze the effectiveness and robustness of the proposed method from various perspectives. We investigate the trade-offs between exploration and exploitation by analyzing the effects of population size and momentum, and also show that our algorithm can produce better representation diversity, which is advantageous to the generalization of DNNs.

2 Background and Related Work

Preliminaries of Lexicase Selection

Lexicase selection is initially proposed as a parent selection method in population-based stochastic search algorithms such as genetic programming (Helmuth et al., 2014; Spector, 2012). Follow-up work has shown that lexicase selection can effectively improve behavioral diversity and the overall performance and on a variety of genetic programming problems (Helmuth et al., 2016; Helmuth & Spector, 2015; Helmuth et al., 2014; Liskowski et al., 2015). The lexicase selection algorithm is outlined in Alg. 1.

The key idea in lexicase selection is that each selection event consider a randomly shuffled sequence of test cases. With the specific ordering, only individuals whose error is minimal among all the already-considered cases are allowed to survive. Moving forward through the sequence of cases, the selection is done when there is only one candidate left or until all the test cases have been gone through, in which case we select randomly from the remaining candidates. Since the ordering of test cases is randomized for each selection event, every test case gets the opportunity to be prioritized

when being put at the beginning of the sequence. As a result, lexicase selection sometimes selects specialist individuals that perform poorly on average but perform better than many individuals on one or more other cases.

Unlike methods such as tournament selection that use a single fitness value and thus tend to always select generalist individuals that have good average performance, lexicase selection does not base selection on an aggregated measure of performance. Such a difference allows lexicase selection to maintain higher population diversity by prioritizing different parts of the dataset during each selection event through the ordering of the cases. It has been shown empirically on a number of program synthesis benchmark problems that lexicase selection substantially outperforms standard tournament selection and typically maintains higher levels of diversity (Helmuth et al., 2016).

In a more general context, lexicase selection can be used in any case where a selection procedure occurs with regard to performance assessment of multiple candidates with a set of test cases. Recent work also explores the usage of lexicase selection in rule-based learning systems (Aenugu & Spector, 2019), symbolic regression (La Cava et al., 2016), machine learning (La Cava & Moore, 2020b;a), and evolutionary robotics (Huizinga & Clune, 2018; La Cava & Moore, 2018). In this work, we aim to explore the effectiveness of lexicase selection in the context of deep learning optimization from the perspective of improving the diversity of gradient-based representation learning for better generalization. While there are also other parent selection methods (Fieldsend & Moraglio, 2015; Galvan-Lopez et al., 2013; Krawiec & Liskowski, 2015) that have been proposed to achieve similar goals, in this work we focus on investigating the usage of lexicase selection in deep learning. Further discussion of comparisons between lexicase selection to other selection methods are out of the scope of this work.

Deep Neuroevolution

While backpropagation (Rumelhart et al., 1985) with gradient descent has been the most successful method in training DNNs with fixed-topology in the past few decades (LeCun et al., 2015), there are also attempts to train DNNs through evolutionary algorithms (EAs). Such et al. (2017) proposed a gradient-free method to evolve the network weights by using a simple genetic algorithm, and was able to evolve a relatively deep network (with 4 million parameters) and demonstrated competitive results on several reinforcement learning benchmark problems. Recent work (Pawełczyk et al., 2018) did a pilot study on combining simple GA schema with gradient-based learning, where gradient training is used as part of the mutation process. Although the method was tested only on one dataset, the results were encouraging and offered some insights on combining GAs with gradient learning. Following this trend, this paper aims to use a popular algorithm in genetic programming, lexicase selection, to improve the network generalization by treating image recognition as an uncompromising problem.

From a broader perspective that is also closely related to this work, there has been a surge of interest in methods for Neural Architecture Search (NAS) (Elsken et al., 2019), where evolutionary algorithms gain high popularity. A majority of methods (Floreano et al., 2008; Liu et al., 2017; Miikkulainen et al., 2019; Real et al., 2019; 2017; Stanley & Miikkulainen, 2002; Xie & Yuille, 2017) use EA to search neural network topologies and use backpropagation to optimize network weights, and some others (Stanley & Miikkulainen, 2002) use EA to co-evolve topologies along with weights. While our method can be easily extended to the NAS problem, this work focuses on training various fixed-topology networks in order to make fair comparisons to show the significance of using lexicase selection to improve model generalization.

3 Gradient Lexicase Selection

Our goal is to integrate the idea of lexicase selection to improve the generalization of DNNs, and at the same time to the greatest extent keep the efficiency and effectiveness of the popular gradient-based learning. We propose Gradient Lexicase Selection as an optimization method to combine the strength of these two methods. An overview of the algorithm is demonstrated in Alg. 2. We describe its components in details in this section as follows.

Algorithm 2: Gradient Lexicase Selection

Data:

- `data` - the whole training dataset
- `candidates` - set of p instances of the DNN model initialized with the same parameters

Result:

- an optimized DNN model

```
epoch ← 0
while True do
    subsets ← p equal-size subsets obtained through random sampling from the
      entire data without replacement
    Use gradient descent and backpropagation to optimize each of the p
      candidates on each of the p subsets respectively
    cases ← randomly shuffled sequence of data to be used in lexicase selection
    candidate_ids ← number set of {0, 1, 2, ··· , p − 1}
    while True do
        Evaluate all the candidates that have their ids in candidate_ids on
          the first case in cases.
        if at least one candidate makes the correct prediction then
            candidate_ids ← the subset of the current candidate_ids that the
              corresponding candidates make the correct prediction
        else
            candidate_ids ← the subset of one randomly sampled id from the
              current candidate_ids
        end
        if candidate_ids contains only one single id then
            parent ← the corresponding candidate in candidates with the
              remaining id in candidate_ids
            break the inner loop
        end
        if cases contains only one single test case then
            parent ← the corresponding candidate in candidates with the
              remaining id in candidate_ids
            break the inner loop
        end
        Remove the first case from cases.
    end
    if epoch < total number of training epochs then
        candidates ← set of p instances of the DNN model copied with the same
          parameters as parent
        epoch += 1
    else
        return parent
    end
end
```

3.1 Evolution of DNNs

First, we introduce the general evolutionary framework that uses a combination of gradient descent and evolution. Given the network topology, we first initialize all the parameters $\mathbb{W}_0$ as the initial parent weight. For each generation, given a population size of p, we generate p instances of the model as p offspring with the same weight as the parent weight, namely, $\mathbb{W}^{(0)} = \mathbb{W}^{(1)} = \cdots = \mathbb{W}^{(p-1)} = \mathbb{W}_0$. We then perform mutation on these offspring and use lexicase selection to select the parent for the next generation.

3.2 MUTATION BY SUBSET GRADIENT DESCENT

For each generation, we have p instances of the model starting with the exact same weights. Instead of random mutations such as adding gaussian noise as commonly used in neuroevolution, we propose a gradient-based mutation method called subset gradient descent.

Given the whole training dataset $\mathbb{S}_{train}$, we divided it into p subsets with random sampling, as $\mathbb{S}_{train}^{(0)}, \mathbb{S}_{train}^{(1)}, \cdots, \mathbb{S}_{train}^{(p-1)}$. We then train each model instance accordingly on one of the subsets using the normal mini-batch stochastic gradient descent. The mutation is done when all the training data is consumed, which is one epoch in traditional deep learning.

There are several advantages of the proposed mutation method. First, since all the offspring are trained with different non-overlapping training samples, they are likely to evolve diversely, especially when data augmentation is also included. Secondly, each off-spring is trained using gradient descent, meaning they will be optimized efficiently towards the objective, comparing to random mutation methods such as gaussian noise. Thirdly, if implemented with distributed training, all the offspring can be trained simultaneously to further reduce computation time. In general, the subset gradient descent aims to find a balance between exploration and exploitation toward the whole evolution process.

3.3 LEXICASE SELECTION FOR DNNS

After mutation, the offspring become candidates and we use lexicase selection to select a parent from them for the next generation. First, a randomly shuffled sequence of training data points (without data augmentation) is used as the 'test' cases for the selection purpose. Starting from the first training sample, we evaluate all the candidates on each case individually and remove the candidate from the selection pool if it does not make the correct prediction. This process is repeated until if 1) there is only one instance left, which will be selected as the parent for the next generation, or 2) all the training samples are exhausted and more than one candidates survive, in which case we randomly pick a candidate from the selection pool.

For the selection process, we do not hold out another validation set because 1) if we choose to use a validation set, the validation set should have an adequate size in order to ensure its diversity and generality, which means the training set will be noticeably smaller, and thus the training performance is likely to degrade; 2) since each model instance only gets access to part of the (augmented) training data, the selection performed on the original training data is still effective, since the exact same data was never used in the mutation (training).

An important feature of lexicase selection is that it treats all the cases equally and thus there is no way to modify its selection pressure. The motivation behind lexicase selection is to allow the survival of those models which may not perform best overall but were able to solve the given testing cases. Such a model is likely to learn essential feature representations that allow it to make correct predictions on specific cases where all others fail. By letting lexicase selection guide the training process, the neural network can potentially learn more diverse representations that finally contribute to better generalization.

To better accommodate the situation of training deep models on large-scale datasets, we also make some slight modifications to the original lexicase selection algorithm in regard to the tie situations, *i.e.*, when all the remaining candidates fail to make the correct prediction on one case. Original lexicase selection lets all the candidates survive because they all have the same "best" performance, which is failure. However, in the early stages of DNN training, almost all the candidates fail straightly in the first few cases, but the original lexicase selection will proceed to evaluate them until someone happens to get a correct prediction by chance, which is inefficient especially on large datasets. So we modify the algorithm to randomly select a candidate from the remaining candidates if they all fail. The modification improves the efficiency of early-stage training and has not shown any noticeable effect on the final model performance.

4 EXPERIMENTAL RESULTS

The proposed Gradient Lexicase Selection is tested on the task of image classification, which is one of the most common benchmark problems in computer vision and deep learning in general. We implement the algorithm on six popular DNN architectures (VGG (Simonyan & Zisserman, 2015), ResNet (He et al., 2016), DenseNet (Huang et al., 2017), MobileNetV2 (Sandler et al., 2018), SENet (Hu et al., 2018), EfficientNet (Tan & Le, 2019)). To show the significance of our method, we also implement the original momentum-SGD training as baselines for all the architectures.

Three benchmark datasets (CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), and SVHN (Netzer et al., 2011)) are used for evaluation. These datasets comprise 32×32 pixel real-world RGB images of common objects (CIFAR-10, CIFAR-100) and street scene digits (SVHN). The training is done using the training set only and we evaluate the methods on the test set after the training process is finished. Note that for illustration purposes, we only use the training dataset of SVHN without the large *extra* set, so the results are not comparable to other work.

4.1 IMPLEMENTATION DETAILS

Each network architecture with baseline SGD training and its corresponding counterpart with gradient lexicase selection are trained with identical experimental schemes. We follow standard practices and perform data augmentation with random cropping with padding and perform random horizontal flipping during the training phase (no augmentation is used during selection phase). The input images are normalized through mean RGB-channel subtraction for all the phases. For both baseline and lexicase, we use SGD with momentum of 0.9. For lexicase, we use the Reset Momentum option that re-initialize the momentum parameters for each epoch, which is explained in detail later in Sec. 5.2.

The batch size is set to 128 for CIFAR-10 and 64 for CIFAR-100 and SVHN. The initial learning rate is set to 0.1 and tuned by using Cosine Annealing (Loshchilov & Hutter, 2017).

We set the total number of epochs as 200 for baseline training and as $200(p+1)$ for gradient lexicase selection, where p is the size of population. For each epoch in lexicase, the mutation only uses $1/p$ of the training data to train each model instance, so we keep the total iterations of weight update of lexicase training similar to baseline training to ensure convergence.

4.2 IMAGE CLASSIFICATION RESULTS

The image classification results are shown in Tab. 1. We report the mean percentage accuracy (*acc.*) with standard deviation (*std.*) obtained by running the same experiment with three different random seeds. The last column (*acc.* $\uparrow$) calculates the difference of accuracy by using our method compared to baseline, where positive numbers indicate improvement. We can first see that by using our method, most of the architectures gain a significant improvement on the testing result. On the easier SVHN dataset, we can still observe moderate and consistent improvement. To show the robustness of our algorithm, we use the same population size of 4 for lexicase in all the experiments, meaning the performance may be further improved if extra tuning is performed. The ablation study on the effect of population size is described later in Sec.5.1.

Beyond those improvement, we also find that among all the architectures, our method surprisingly fails to improve MobileNetV2 on both CIFAR-10 and CIFAR-100. The main difference between MobileNetV2 and other architectures is that it is a highly optimized architecture with over a magnitude less parameters comparing to other architectures. Sandler et al. (2018) stated that they tailor the architecture to different performance points, which can be adjusted depending on desired accuracy/performance trade-of. As a result, it is likely that the accuracy is restricted by the model size, and even with better training strategies the performance is not going to improve. Such results indicate that our method may not work directly with architectures that have been optimized by using other training methods. But for other more general architectures our method work directly out-of-the-box without further tuning.

Table 1: Image classification results. We report the mean percentage accuracy (*acc.*) with standard deviation (*std.*) obtained by running the same experiment with three different random seeds. The last column (*acc.* ↑) calculates the difference of accuracy by using our method compared to baseline, where positive numbers indicate improvement.

Dataset	Architecture	Baseline		Lexicase		
		acc.	*std.*	*acc.*	*std.*	*acc.* ↑
CIFAR-10	VGG16	92.85	0.10	93.40	0.13	**0.55**
	ResNet18	94.82	0.10	95.35	0.06	**0.53**
	ResNet50	94.63	0.46	94.98	0.18	**0.34**
	DenseNet121	95.06	0.31	95.38	0.04	**0.32**
	MobileNetV2	94.37	0.19	93.97	0.12	-0.39
	SENet18	94.69	0.14	95.37	0.23	**0.68**
	EfficientNetB0	92.60	0.18	93.00	0.22	**0.40**
CIFAR-100	VGG16	72.09	0.52	72.53	0.20	**0.44**
	ResNet18	76.33	0.29	76.68	0.40	**0.35**
	ResNet50	76.82	0.96	77.44	0.25	**0.63**
	DenseNet121	78.72	0.82	79.08	0.26	**0.36**
	MobileNetV2	75.87	0.28	75.57	0.30	-0.30
	SENet18	76.97	0.06	77.22	0.29	**0.25**
	EfficientNetB0	71.03	0.86	71.36	0.87	**0.33**
SVHN	VGG16	96.27	0.06	96.29	0.08	**0.02**
	ResNet18	96.43	0.14	96.62	0.08	**0.19**
	ResNet50	96.69	0.21	96.74	0.07	**0.04**
	DenseNet121	96.82	0.16	96.87	0.03	**0.05**
	MobileNetV2	96.23	0.13	96.26	0.07	**0.03**
	SENet18	96.62	0.19	96.59	0.11	-0.03
	EfficientNetB0	96.14	0.12	95.94	0.10	-0.20

5 Ablation Studies

In this section, we design several ablation studies to further analyze and validate the effectiveness of the proposed method. Unless specifically mentioned, all the implementation details follows the same practices in Sec. 4.1.

5.1 Population Size

Population size is an essential hyperparameter for evolutionary algorithms. In our case, the population size of DNNs has more constraints such as computation cost and GPU memory, so it has to be much smaller comparing to those for GP problems. As a result, we test lexicase gradient selection with population sizes of $2, 4, 6, 8$. For illustration purposes, two architectures (VGG16 and ResNet18) are evaluated on the CIFAR-10 dataset. The results are shown in Tab. 2.

First, we can see that lexicase is relatively robust to different population size p. Under all the population size configurations lexicase manages to outperform baseline significantly. Since there have not been a trend of increased accuracy with larger population size, the generalization performance does not seem to increase with more population. This observation aligns well with the behavior of lexicase selection in GP problems (Helmuth et al., 2018; La Cava et al., 2019), where there seems to be an optimal population size through the trade-off between exploration and exploitation. For larger population size, the individuals may not evolve different-enough behaviors from each other, and thus the diversity of population may become lower. Besides, larger population size also requires more computation and GPU memory in deep learning. Fortunately, we find that even with relatively small population size, we can already get good results in gradient lexicase selection.

Table 2: Comparing different population sizes on CIFAR-10. Lexicase is relatively robust to different population size p, and it manages to outperform baseline with all the configurations in this experiment. The generalization performance does not seem to increase with larger population size.

Architecture	Baseline	Lexicase with population size p			
		$p = 2$	$p = 4$	$p = 6$	$p = 8$
VGG16	92.85	93.61	93.40	**93.92**	93.37
ResNet18	94.82	**95.50**	95.35	95.27	95.38

Table 3: Comparing different momentum configurations on CIFAR-10. Resetting momentum after each selection event avoids too much aggregation of gradients, which results in a higher diversity of offspring and thus better generalization performance.

Architecture	Baseline	Lexicase with different momentum options		
		No Momentum	Reset Momentum	Inherit Momentum
VGG16	92.85	92.95	**93.40**	93.13
ResNet18	94.82	94.77	**95.35**	95.23

5.2 MOMENTUM

In deep learning, SGD with momentum (Sutskever et al., 2013; Liu et al., 2020) has been one of the most widely adopted method for training DNNs. Momentum accelerates gradient descent by accumulating a velocity vector in directions of persistent reduction in the objective across iterations. This accumulating behavior actually interferes with the proposed gradient lexicase selection algorithm, because model instances in the population get different gradient updates, and thus will have different momentum parameters.

To solve this issue, we propose three options: 1) No Momentum: do not use momentum at all; 2) Reset Momentum: use high momentum rate and re-initialize the momentum parameters every epoch for each model instance; 3) Inherit Momentum: when selecting the parent model, also copy the momentum parameters along with the model parameters to all the instances in the next generation. For this study, we also test two architectures (VGG16 and ResNet18) with population size of 4 on the CIFAR-10 dataset. The results are shown in Tab. 3.

The key idea of lexicase selection is to evolve offspring with different behaviors, and make selection to prioritize one during each iteration. In this way, the population can maintain a high diversity. On the other hand, the way how momentum works tends to find an aggregated direction of gradient update accumulated through time. From the results, we can see that the Reset Momentum option works the best, indicating that if we inherit momentum, it will influence the mutation over generations and thus the selection strength of lexicase will be negatively affected. By resetting momentum each epoch, only the mutation in the current generation is accelerated by using momentum SGD, which results in a higher diversity of offspring. In general, the momentum options can also be viewed as different trade-offs of exploration and exploitation.

5.3 REPRESENTATION DIVERSITY

While quantitative results have shown that the proposed method manage to improve the generalization of DNNs, we would like to further investigate the reasons behind for better understanding of the behavior of the algorithm. One hypothesis is that since lexicase selection is able to increase the diversity and generality of the population in GP, it may as well help DNNs learn more diverse representation, which improves the overall model generalization. To validate this, we analyze and compare the feature representations in ResNet-18 trained using normal SGD and gradient lexicase selection. We take the first 100 samples from the CIFAR-10 test set and use global max pooling to obtain the channel-wise activations of `conv_4x` and `conv_5x` layers (as defined in He et al. (2016)). Fig. 1 shows the results.

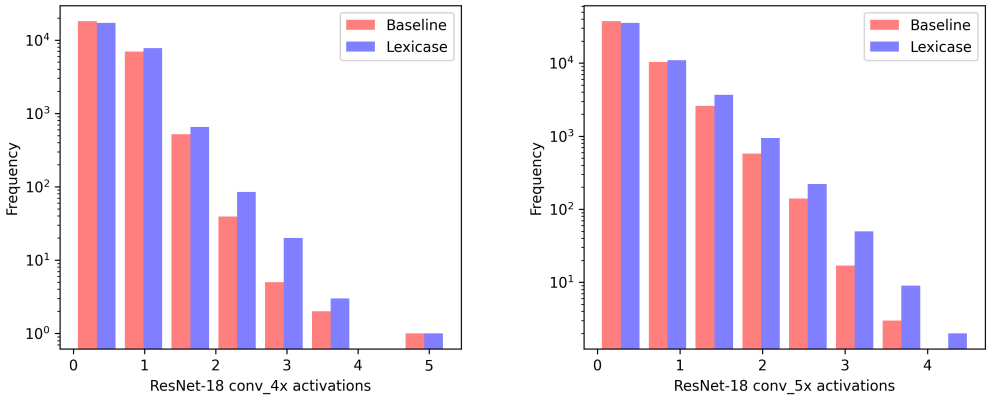

Figure 1: Comparing representation diversity of normal SGD (Baseline in red) and gradient lexicase selection (lexicase in blue). The flatter distribution shows that our method produce more diverse representations.

We can observe that our method produce a flatter distribution of activations with less frequency on 0s and more frequency on other values. Ioffe & Szegedy (2015); Wu & He (2018) shows that normalized distribution of layer activations can help reduce the internal covariance shift of DNNs during training, and thus improves the training efficiency and model generalization. Similarly, our method manages to learn more diverse representations by incorporating lexicase selection into training framework, which is advantageous to the generalization of DNNs.

6 Conclusion and Future Work

In this work, the adoption of lexicase selection in optimizing deep neural networks is explored. We propose gradient lexicase selection, an evolutionary algorithm that incorporates lexicase selection with stochastic gradient descent to help DNNs learn more diverse representations for better generalization. Experimental results show that the proposed method can improve the performance of several popular DNN architectures on benchmark image classification datasets. Several ablation studies further validate the robustness and advantages of our method from different perspectives. More specifically, we investigate the trade-offs between exploration and exploitation by analyzing the effects of population size and momentum. We also show that our algorithm can produce better representation diversity, which is advantageous to the generalization of DNNs.

As described in Sec. 4.2, the current lexicase may not work with architectures that have been highly optimized, indicating a potential correlation between network architecture and lexicase selection. For future directions, we look forward to explorations on how lexicase can be used in optimizing neural architectures along with the parameters, and the integration of lexicase in neural architecture search in general.

Reproducibility Statement

We submit our source code as the supplementary material for the review process, which can be used to reproduce the experimental results in this work. The configurations and implementation details are described in Sec. 4.1.

References

Sneha Aenugu and Lee Spector. Lexicase selection in learning classifier systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 356–364, 2019.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Jonathan E Fieldsend and Alberto Moraglio. Strength through diversity: Disaggregation and multi-objectivisation approaches for genetic programming. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 1031–1038, 2015.

Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary intelligence*, 1(1):47–62, 2008.

Edgar Galvan-Lopez, Brendan Cody-Kenny, Leonardo Trujillo, and Ahmed Kattan. Using semantics in the selection mechanism in genetic programming: a simple method for promoting semantic diversity. In *2013 IEEE Congress on Evolutionary Computation*, pp. 2972–2979. IEEE, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Thomas Helmuth and Lee Spector. General program synthesis benchmark suite. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 1039–1046, 2015.

Thomas Helmuth, Lee Spector, and James Matheson. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation*, 19(5):630–643, 2014.

Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. Lexicase selection for program synthesis: a diversity analysis. In *Genetic Programming Theory and Practice XIII*, pp. 151–167. Springer, 2016.

Thomas Helmuth, Nicholas Freitag McPhee, and Lee Spector. Program synthesis using uniform mutation by addition and deletion. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1127–1134, 2018.

John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Joost Huizinga and Jeff Clune. Evolving multimodal robot behavior via many stepping stones with the combinatorial multi-objective evolutionary algorithm. *arXiv preprint arXiv:1807.03392*, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Krzysztof Krawiec and Paweł Liskowski. Automatic derivation of search objectives for test-based genetic programming. In *European Conference on Genetic Programming*, pp. 53–65. Springer, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Tech. Report*, 2009.

William La Cava and Jason Moore. Behavioral search drivers and the role of elitism in soft robotics. In *ALIFE 2018: The 2018 Conference on Artificial Life*, pp. 206–213. MIT Press, 2018.

William La Cava and Jason H Moore. Genetic programming approaches to learning fair classifiers. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 967–975, 2020a.

William La Cava and Jason H Moore. Learning feature spaces for regression with genetic programming. *Genetic Programming and Evolvable Machines*, 21(3):433–467, 2020b.

William La Cava, Lee Spector, and Kourosh Danai. Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 741–748, 2016.

William La Cava, Thomas Helmuth, Lee Spector, and Jason H Moore. A probabilistic and multi-objective analysis of lexicase selection and ε-lexicase selection. *Evolutionary Computation*, 27 (3):377–402, 2019.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Pawel Liskowski, Krzysztof Krawiec, Thomas Helmuth, and Lee Spector. Comparison of semantic-aware selection methods in genetic programming. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 1301–1307, 2015.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017.

Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum. *arXiv preprint arXiv:2007.07989*, 2020.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=Skq89Scxx`.

Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing*, pp. 293–312. Elsevier, 2019.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Krzysztof Pawełczyk, Michal Kawulok, and Jakub Nalepa. Genetically-trained deep neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 63–64, 2018.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pp. 2902–2911. PMLR, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Lee Spector. Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pp. 401–408, 2012.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Lingxi Xie and Alan Yuille. Genetic cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1379–1388, 2017.