PHYLOGENETIC REPLAY LEARNING IN DEEP NEURAL NETWORKS

Jean-Patrice Glafkides, Gene I Sher, Herman Akdag

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 08, No. 03, September 2022.

PHYLOGENETIC REPLAY LEARNING IN DEEP NEURAL NETWORKS

Jean-Patrice Glafkides[1], Gene I. Sher[2] and Herman Akdag[1]

(Received: 31-Jan.-2022, Revised: 18-Apr.-2022, Accepted: 22-Apr.-2022)

ABSTRACT

Though substantial advancements have been made in training deep neural networks, one problem remains, the vanishing gradient. The very strength of deep neural networks, their depth, is also unfortunately their problem, due to the difficulty of thoroughly training the deeper layers due to the vanishing gradient. This paper proposes "Phylogenetic Replay Learning", a learning methodology that substantially alleviates the vanishing-gradient problem. Unlike the residual learning methods, it does not restrict the structure of the model. Instead, it leverages elements from neuroevolution, transfer learning and layer-by-layer training. We demonstrate that this new approach is able to produce a better performing model and by calculating Shannon entropy of weights, we show that the deeper layers are trained much more thoroughly and contain statistically significantly more information than when a model is trained in a traditional brute force manner.

KEYWORDS

Neural networks, Neuroevolution, Phylogenetic replay learning, Deep learning, Vanishing gradient.

1. INTRODUCTION

Nature evolved the nervous system through eons of trial and error, from the first apparition of the neuronal cell to the complex brains we possess today. The field of machine learning has made tremendous progress during the past decade, predominantly owing to the improvement of CPU performance, data accessibility, optimization of deep neural network (DNN) algorithms, but also just as significantly due to the improvements in hardware and the use of GPUs. Artificial neural networks are called deep when they have more than 3 layers of neurons (though some categorize DNNs as those having more than 9 layers) and are capable of being tuned to reach a specific goal through the use of an optimization algorithm, mimicking the role of synaptic plasticity in biological learning. This approach has led to the emergence of highly efficient algorithms that are capable of learning and solving complex problems [1]. Two of the main limitations of such algorithms are: 1. Their topologies are built empirically and 2. Due to the depth of deep neural networks, they are affected by the vanishing-gradient problem. Though this paper primarily concentrates on solving the 2nd problem (the vanishing-gradient problem), we demonstrate its use by applying it to a model that was evolved through neuroevolution.

We do this because:

1. In the last few years, substantial advancements have been made in automated model search and construction. These automated model construction and model search methods are commonly called neuroevolutionary methods, due to the use of evolutionary algorithms to search for optimal model architectures [2]. These methods have demonstrated a strong ability to produce state-of-the-art models demonstrating excellent results in numerous domains [3]-[5] with very surprising results in some cases [6]-[7]. Several works exploring the use of evolutionary computation in deep network optimization [8]-[10] were produced.

2. Our new proposed method, Phylogenetic Replay Learning (PRL), can be perfectly combined with both, traditional, but also neuroevolutionary methods to leverage the ability to construct deep and complex networks from simple ones.

It must be noted that the objective of this paper is not to discuss or compare any specific model search or neuroevolutionary method, like EANT1/2 [11], CoSYnE [12], DXNN or NEAT, efficiency over other methods that have already been addressed [13]-[15], but to explore the use of backpropagation training in pre-planned mutations, training layers one at a time as the deep neural network is constructed. With all the accomplishments of deep learning, it remains difficult to build models that generalize or adapt

1. J.-P. Glafkidès (ORCHID: 0000-0001-5273-9948) and H. Akdag are with PARAGRAPHE EA 349 - PARIS VIII University, France. Emails: jp@glafonline.com and herman.akdag@univ-paris8.fr

2. G. I. Sher (ORCHID: 0000-0002-2086-4370) is with DataValoris - WY, USA. Email: gene@datavaloris.com

efficiently to complex-problem domains and data. One of the bigger difficulties being faced when building complex and deep models that converge correctly is the vanishing-gradient problem [16]-[18] which is yet to be solved [19]. It is this problem, the vanishing gradient, that the PRL approach is also aimed at solving. With the increasing number of layers that are used, the vanishing-gradient problem can cause the gradient to become too small for effective weight parameter updating. This is due to certain activation functions, like the sigmoid function, which squashes a large input space into a small one between 0 and 1. Thus, a large change in the input of the sigmoid function will cause a small change in the output and with it the derivative also shrinks. This problem is exacerbated with deeper layering; the gradient decreases exponentially as we propagate down to the initial layers. A small gradient means that the weights and biases of the initial (deeper) layers will not be trained effectively. Since these initial layers are often crucial to recognizing the core elements of the input data, this can lead to overall inability of the whole network to learn effectively. This effect can be partially mitigated by using other activation functions, such as relu for example. Other ways of combating this problem are specific architectures, like the residual neural network [20] which attempts to decrease the effect of this problem by connecting deeper layers directly to the output. However, it is not enough and too restrictive. This calls for the development of new methods specifically designed to enhance learning capabilities and counter the vanishing-gradient effect. A method is needed that will not restrict us to the use of specific neural topologies or activation functions.

The objective of this paper is to compare the performance of training a DNN all at once, *versus* training it one mutation at a time as a pre-planned model is being constructed (PRL training) and demonstrate that the latter produces a better outcome, with each layer of such model storing statistically greater amount of information. The Phylogenetic Replay Learning (PRL) requires a trace of model's complexification, from a simple shallow version to the final complex DNN. When this trace is available, it performs re-training of the layers as it adds layer on-top of layer within the trace. This iterative re-training approach ensures that every layer was at some point the output layer (or close to it) and thus was affected by the gradient descent learning algorithm to a greater extent, while the deeper layers were "re-tuned" to work effectively in the deeper model. When this approach is combined with neuroevolution, the system first evolves the final model from a simple initial seed model while also building its trace of mutations (which new layers are added on top of which or which layer is changed or get linked to others) and then it re-traces those evolutionary steps (the phylogeny), while re-training the model at every evolutionary step, as shown in the Figure 1. In the following sections, we will discuss in detail the PRL method. First, we will cover the background of the pertinent domains, neuroevolution and the vanishing-gradient problem. We will then provide definitions of the terms used in this paper. In the methods section, we provide a detailed PRL algorithm. In the results section, we will present the experiments performed and their results. Finally, we will conclude with the analysis and discussion of the results achieved.

2. BACKGROUND

2.1 The Vanishing-gradient Effect (VGE)

The most common neural network (NN) optimization algorithm is based on the use of stochastic gradient descent. This involves first calculating the prediction error made by the model and then using the error to estimate a gradient used to update each weight layer by layer, cascading backwards in the network. This error gradient is propagated backward through the network from the output layer to the input layer, updating the weights to minimize the difference between the actual NN output and the expected output. It is useful to train NNs with many layers. The addition of deeper layers increases its capacity, making it capable of learning more complex mapping functions between input and output when a large training dataset is provided. A problem with training networks with many layers (e.g. deep neural networks) is that the gradient diminishes dramatically as it is propagated backward through the network. The error may be so small by the time it reaches layers close to the input of the model that it may have very little effect. Thus, this problem is referred to as the “vanishing-gradient” problem.

2.2 Neuroevolution

Neuroevolution is a machine-learning technique that applies an evolutionary algorithm to construct artificial NNs, taking inspiration from the biological evolutionary process.

2.3 Definitions

Champion: is an NN model (topology and weights) representing the best model that neuroevolution is able to produce to solve a problem.

Direct Deep Learning (DDL): is what we call the standard/default training of a model using backpropagation (Adam, QProp, …etc.) to differentiate it from the PRL method. It is a method that is applied to the DNN without the use of neuroevolution or PRL. In our experiments, the training algorithm used in the framework was set to Adam. The DDL is also known as end-end training

Hall of Fame (HOF): or HOF for short, is a list our neuroevolutionary system holds of the best performing agents/models. In our tests, HOF was set to size 10,

Initial Model: is the seed model used as the starting point of model search in neuroevolution.

Mutations: at each step of the evolutionary process, we apply mutation(s) to the topology of the parent in order to create an offspring. A topological mutation can add a layer to the model, mutate existing layer's parameters, remove a layer, clone an existing layer, add or change a link between two layers or swap one layer for another type of layer.

Phylogenetic Replay Learning (PRL): is a method of training a model for a specific problem using pre-recorded mutation path of a seed model topology (system implemented and presented in this paper), but doing so one mutation step at a time, following that model's phylogenetic path. In other words, we re-train the model after every applied mutation step once we know what the best model is and what mutation steps were taken to achieve it from the seed model, usually following the path of model complexification from the initial neuroevolution phases. This method is the topic of this paper.

Selection Process: is the mechanism by which the algorithm selects the best entities according to their score (fitness function) and stores them in the "Hall of Fame" (HOF) list.

2.4 Other Methods to Reduce Vanishing-gradient Effect (VGE)

Several other approaches can be used to reduce the VGE, but none are perfect. Using PRL does not preclude one from leveraging other methods as well.

- Activation functions, such as relu for example [21].
- Normalized initialization layers [22]-[23] and intermediate normalization layers [24], which enable networks with tens of layers to start learning/converging with stochastic gradient descent (SGD) with backpropagation [25].
- Specific architectures like the residual neural networks which attempt to decrease the effect of this problem by using pass-through links [20].
- Regularizing deep neural networks by noise injects noise during the training procedure, adding or multiplying noise within the hidden units of the NNs [26].
- Deep cascade learning method proposes a solution to alleviate the VGE [27] by training deep networks in a cascade-like or bottom-up layer-by-layer manner. It reduces the VGE, but was not shown to be better than DDL.

2.5 Metrics

The metrics we use for model comparison is the test accuracy. Early stopping was applied on the score we want to follow and not used for training. Accuracy is used as the metric. To better understand the difference in the informational density of the models, we calculated their weights' Shannon entropy [28] Equation 1, Equation 2 after training.

$$P_i = \frac{i}{\sum_{i=1}^{n}(i)} \tag{1}$$

H entropy:

$$H = -\sum_{i=1}^{n} P_i \ln(P_i) \tag{2}$$

2.6 Dataset

PRL was tested on the 4 "original" datasets from Keras site: MNIST, Fashion MNIST, CIFAR 10 and Tiny Imagenet. CIFAR10 was converted into grayscale with images reshaped to 28*28 pixels, to not only match the same shape as those within MNIST, but also to make it much more complex to learn.

Tiny Imagenet 200 dataset has been chosen to test the system on a more modern, bigger and more difficult to learn dataset. This paper's aim is to compare the PRL method to the standard approach. Thus, the goal of this work is to show that on average, this training approach produces better performing models, with more densely packed information, than the direct approach, by alleviating the VGE. Thus, we believe that for these preliminary results, it is appropriate to use these datasets.

2.7 Tools

We selected tools like Keras that provides the training framework and Raise solution from DataValoris that provides the evolutionary part of the experiments on top of Keras. They have accelerated our work as their engine already provides the unrestricted topological search-based deep-learning neuroevolution. The PRL (recording and replay training) was developed by us for the purpose of this work and presentation of experiments and their results in this paper. The method could be as easily used with other neuroevolutionary systems, like NEAT, EANT1/2, DXNN or GNARL, as long as we record the phylogenetic path of mutations that can then be used to replay the mutations and train the model one step at a time. Finally, all of our experiments were performed on a server with an Nvidia Tesla v100 GPU card. Part of this work was granted access to the HPC/AI resources of IDRIS under the allocation 2021- AD011012674 made by GENCI.

2.8 Seed Model

Table 1 shows the simple model used as the seed model. It includes 7850 parameters and 1 hidden layer in a sequential architecture.

Table 1. Initial model test 1.

Layer Number	Type	Output	Params
1	InputLayer	N, 28, 28, 1	0
2	Flatten	N, 784	0
3	Dense	N, 10	7850

2.9 Selection Rules

During the building of the phylogenetic path, the neuroevolutionary process uses selection based on the score generated by the learning algorithm. The score used as a fitness is the test accuracy of the model. We have set the system such that the learning rate is decreased when the score does not improve for 3 consecutive evaluations. Every generation 10 NNs are trained, then their scores are compared to the NNs in HOF. If a score of an offspring/mutant model within the current generation is higher than that of a model within the HOF that has the same topology, the mutant model replaces the model within the HOF. If the mutant model has the highest score and has a topology not present within the HOF, the model with the lowest fitness within the HOF is removed and the new model is added in its spot.

3. METHODS

PRL is a method to train models using genetically planned mutations over time. It alleviates the vanishing gradient effect through its complete training. The system allows the classical gradient descent method to train each layer, even the very deep ones, more than the traditional learning approach. It does this by retraining each of those layers as the model is being evolved and new layers are added. Each new layer added has the chance of being trained as if it were the first or second layer in the backprop cascade. Frameworks used in this study were the official TensorFlow and plaidML. Datasets have not been augmented during tests. The algorithms were developed in Python. To use the PRL algorithm the experiments have been cut into two phases

3.1 Phase 1: Generating the Champion's Mutation Path through Neuroevolution

PRL requires the existence of the phylogenetic path of the model we need to train. The first phase is meant to build the Champion model while recording its phylogenetic path (mutations that were applied sequentially to generate it). Neuroevolution is used to accomplish (Figure 1) this.

Neuroevolution generates a phylogenetic path (Figure 2) of the best performing model topology aka "champion". In this figure, the champion has 3 ancestors. The figure also shows which

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 08, No. 03, September 2022.

topological mutations were applied to get from one model to the next. The neural architecture used is built by the evolutionary process (could be CNN, Dense layers, Resnet like structure…etc.).

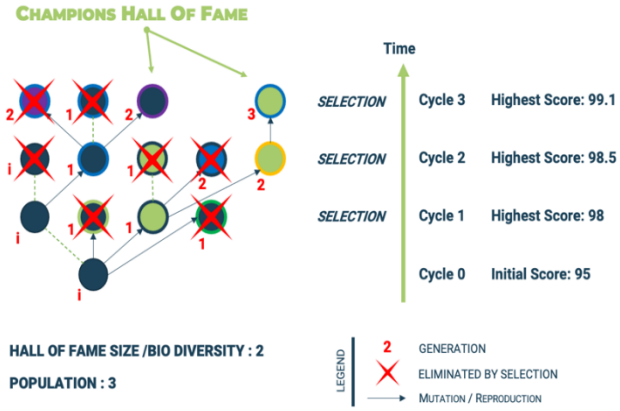

Figure 1. Selection mechanism sample.

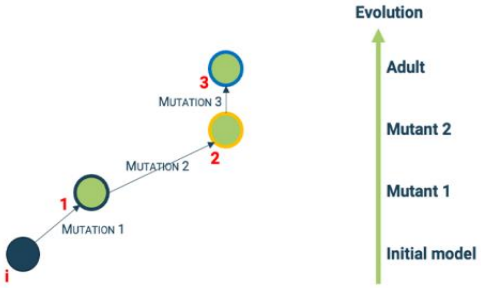

Figure 2. Phylogenetic path of champion.

3.2 Phase 2: Model Generation with the PRL

Now that we have a phylogenetic path that leads to the champion model, we can replay the path from the seed model to champion model (Figure 3).

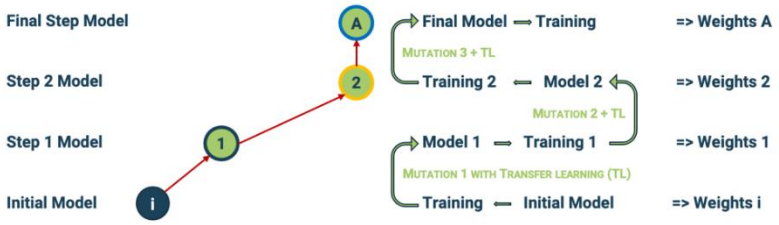

Figure 3. The phylogenetic learning path from initial model to the final one.

When replaying the phylogenetic path, we have to: 1. Generate the seed model with a new set of random synaptic weights and 2. Generate random weights when adding new layers during mutations. This then creates the final model with the same topology as the champion model, but with its own set of parameters. This is a way to statistically include in testing the impact of the initial random weights.

This study was composed of the following steps:

- Phylogenetic path recording: First, an initial simple seed model is trained on a dataset. Using the neuroevolutionary approach, over multiple generations a more complex and better performing NN architecture is evolved and the evolutionary steps leading from the seed NN to the final architecture are recorded in its mutation trace list. The final architecture is what we call the champion model.
- PRL training evaluation: Having the trace from the initial seed model to the champion, the seed model is re-trained using the PRL method X# of times. Using the PRL method, after the application of each mutation in the mutation trace, the system is retrained. This is done for every mutation step, from seed to champion, without resetting the weights between each mutation/training step (in some sense, similarly to transfer learning). This provided the average performance (average of X # of times) of the same champion topology, but trained using the PRL method.
- Champion model DDL retraining: The champion model was re-initialized with random weights and trained on the dataset X# of times using the standard learning approach. This was done to calculate the average performance of the model trained in the standard manner (to which we refer in this paper as "directly applied deep learning" or DDL), with different initial synaptic weights. The early stop patience and epoch number were set to 9 and 60 to avoid a bias where the DDL might not have enough time to train very deep networks.
- Reproducibility testing: In order to confirm the results and test the reproducibility of the method, we did the experiment more than once and on different frameworks. Another champion was

created and PRL again applied using a new seed model, another framework as well as applying it to the more complex Tiny ImageNet dataset.

- Transferability testing: We were also interested in whether the generated model was generalizable to other problems from the same domain and the difference between DDL and PRL-based methods when it comes to transferability. To evaluate the transferability of the Model using the PRL process, we also tested the same champion on other datasets, by retraining it using DDL and PRL methods.
- Data-storage efficiency testing: We calculated the efficiency of information storage in complex models trained through PRL and compared the results to those trained with DDL.

4. RESULTS OF EXPERIMENT 1

In this section, we will first generate a champion and then store its phylogenetic path. Then, we will replay the recorded mutation path with the resulting statistics and compare them to the DDL results.

4.1 Phase 1: Champion 1 Generation

The experiment was setup as follows:

- When using the neuroevolutionary method, a seed population of 20 random minimalistic models is generated.
- 20 agents are generated during every cycle (by way of mutation) from the best agents within the HOF (with a HOF max size of 10), where the probability of using any one agent as the parent of the mutant offspring being proportional to its relative fitness (accuracy) as compared to other HOF agents.
- This experiment used the MNIST dataset.
- The evolutionary engine applied 1-2 (randomly chosen) mutations to create a mutant offspring model from the parent.

The deep-learning parameters used were as follows: 20 epochs with early stopping based on a patience of 3, where patience is based on the test loss metric.

Point of attention: In this work, we refer to the "number of parents since origin" as the agents' generation number. In classic genetic algorithms, the generation is what in this study we call "cycles", therefore an agent of generation 3 and cycle 8 means that it appeared on the 8th iteration and has 3 ancestors (it could have appeared at minimum between cycles 3 to 8).

From the list of champions generated using the neuroevolutionary method during phase 1, we chose the best one, as shown in Table 2.

Table 2. Champion 1 results' information (MNIST).

Score	Cycle	Generation	Parameters	Nodes	Layers
0.9944	96	19	409158	25	13

The chosen champion has 409158 parameters spread between 25 nodes that are 13 layers deep. It has been generated on the 96th cycle and is generation 19 (it has 19 ancestors). Its score 99.44% is close to state-of-the-art on non-augmented MNIST dataset. The mutations recorded at each step that lead to the final champion topology are displayed in Table 3. At every evolutionary step, 1-2 mutation(s) were applied. The number of mutations applied at each step is limited to a maximum of 2 in order to generate a complex model with small changes between each step, which allows PRL to work on smaller parts during each mutation.

Table 3 presents a base of comparison; it shows PRL scores of the champion NN at each step of its evolutionary path. Those scores have been used as the selection criteria for HOF entrance of the offsprings during the evolutionary process. This first result shows that the model has increased in size. This is a classic behavior of an evolutionary algorithm if no size restrictions are used during model generation and mutation. We also see that the Shannon entropy decreases from generation to generation, from 8.98 to 8.90 (excluded initial model of 12.51).

We can interpret this reduction as the increase in organization and amount of useful information stored

by the model's weights. We move from an almost random set of weights to a set of weights that store useful information, a more organized distribution.

Table 3. Phylogenetic path and scores of the chosen champion.

STEP	SIZE	SCORE	SHANNON	STEP	SIZE	SHANNON	SCORE
0	7850	8.79	12.5151	10	264970	8.92824	99.3
1	94906	97.51	8.98112	11	269130	8.92447	99.27
2	27082	98.43	8.97188	12	300874	8.92426	99.28
3	58538	98.96	8.98559	13	300874	8.91894	99.33
4	90346	99.03	8.98102	14	304970	8.91918	99.34
5	90282	99.03	8.96616	15	304970	8.91798	99.34
6	183818	99.23	8.95914	16	304970	8.91156	99.35
7	183818	99.19	8.9567	17	304970	8.90396	99.36
8	258570	99.24	8.94914	18	405486	8.90111	99.41
9	264330	99.29	8.9393	19	409158	8.90037	99.44

4.2 Phase 2: DDL *versus* PRL Statistics

During phase 2, we gather the result metrics of the two different learning approaches to evaluate the impact of using PRL as compared to DDL.

4.2.1 DDL of Champion 1

To evaluate the learning capacity of the model, we conducted 50 runs using the standard learning method applied directly to the final champion model. The initial weights in each experiment were randomly generated. This number of runs allows us to calculate a statistically relevant standard deviation. In theory, the DDL of the champion model could have the same performance as the original champion (and potentially higher), but the probability that these 409158 random parameters reach an optimum is very low. The more complex and deeper the model, the greater the effect PRL method is expected to produce by countering the vanishing-gradient effect (VGE). To perform these experiments and to maximize the probability of reaching a good local minimum, 60 epochs per run were used, with patience set to 6. During our experiments, a maximum of 53 epochs were used before early stoppage occurred. An average of 45 epochs out of 60 were used before early stoppage was triggered. During phase 1 of the PRL method, the champion achieved an accuracy of 99.44%. Its Shannon entropy is 8.90037. The best score/accuracy achieved using DDL of the champion model was 99.05%, with a statistically significant difference (Table 4). We suspect that the VGE is the root cause of this result. Furthermore, we can also see that Shannon entropy of the best performing model trained using the standard approach (9.1227) is also higher than the entropy of the champion model produced during phase 1 of the PRL method.

Table 4. Applying DDL to the champion model (MNIST).

	SCORE	SHANNON
BEST	99.05	9.1227
MEAN	98.93	9.1615
Standard Deviation	0.067	0.0168

We see that the application of DDL to the model is also less efficient than that produced through phase 1 of the PRL method.

4.2.2 Phylogenetic Replay Learning

From initial model, the mutations are applied based on the phylogenetic path of the champion model. The weights are randomly generated for the new mutated layers as well as seed model. We reran the PRL experiment 50 times to gather data on which to base our averages. Weights were not reset between mutations (which can be considered as transfer learning). Table 5 shows the results of the 50 PRL experiments.

The best score reached was 99.40% with an average of 99.26%. This score is very close to that of the original champion model, which reached 99.44%. Thus, there is substantial consistency. Table 6 shows statistical information of the DDL and PRL experiments.

Table 5. PRL of the champion model (MNIST).

Step	Mean Score	Std. Deviation	Best Score	Shannon	Step	Mean Score	Std. Deviation	Best Score	Shannon
0	92.14	0.0771	92.33	12.5163	10	99.16	0.0647	99.32	8.8497
1	97.68	0.2711	98.11	8.9256	11	99.17	0.0700	99.35	8.8454
2	98.35	0.0925	98.58	8.9015	12	99.18	0.0563	99.31	8.8437
3	98.88	0.0841	99.02	8.9079	13	99.19	0.0641	99.34	8.8415
4	99.01	0.0676	99.16	8.8960	14	99.19	0.0626	99.34	8.8396
5	99.05	0.0634	99.13	8.8802	15	99.19	0.0618	99.31	8.8373
6	99.12	0.0553	99.23	8.8749	16	99.24	0.0606	99.36	8.8262
7	99.11	0.0541	99.22	8.8702	17	99.24	0.0521	99.37	8.8208
8	99.14	0.0515	99.27	8.8628	18	99.24	0.0542	99.35	8.8185
9	99.14	0.0660	99.26	8.8547	19	99.26	0.0628	99.40	8.8147

Table 6. Statistics of experiments.

	DDL	PRL		
Mean Score	98.93%	99.26%	POOLED VARIANCE	4.2E-07
VARIANCE	4.5E-07	3.9E-07	T STAT	-25.13668
OBSERVATIONS	50	50		

- The scores are lower than those produced by the champion itself (which followed the optimal path). This is probably due to the randomly generated weights during each step. But, we can also see that the standard deviation of the experiments is low, thus there is performance consistency in the results produced by PRL.
- The score produced by PRL is better than that produced by DDL. With an average maximum of 99.26% compared to 98.93% of DDL, the difference is statistically significant ($p < 0.001$ - Table 6) and the distribution is well separated (Figure 4). Similarly, comparing both maximums of 99.40% (PRL) to 99.05% (DDL), we see a statistically significant difference. Giving DDL more time to train (60 epochs) does not improve its performance (early stop almost always occurs before the epoch number).
- The standard deviation of PRL is lower (better) than that of DDL (Table 6). We believe that this confirms that PRL is a more robust approach and more resilient to random weight initialization.
- During the PRL, the Shannon value consistently decreased at every step (Table 5) of the process. This can be seen as an increasing organization/informational density of the model while the model's complexity increases at each step.
- The Shannon entropy of the PRL-based model is lower (better) than that of the DDL-based model; 8.81 *versus* 9.16.

The last two results reinforce the hypothesis that PRL alleviates the VGE. Though more tests must be conducted to further analyze the approach, this preliminary work shows a promising path. Table 7 shows that when using DDL, the Shannon entropy of the last layers in the model is lower than that of those in the PRL-trained model (bold values for lowest entropy in Table 7). Calculated entropy for each layer of champion 1 are displayed for comparison.

Table 7. Comparison of Shannon entropy between layers.

LAYER #	TYPE	DDL	PRL	Ref. Champion
2	CONV2D	9.162	**8.815**	
3	SEPCNV2D	8.372	**8.234**	8.235
4	CONV2D	15.159	**15.138**	15.142
5	DENSE	14.776	**14.727**	*14.715*
6	DENSE	**11.683**	11.687	11.691
7	CONV2D	14.155	**14.081**	*14.054*
8	CONV2D	14.186	**14.077**	*14.068*
9	DENSE	12.104	**12.101**	*12.095*
10	DENSE	**11.684**	11.688	11.689

11	DENSE	**17.405**	17.512	17.517
12	DENSE	**11.689**	11.711	11.713
13	DENSE	**12.466**	12.588	*12.582*

This hints that the standard training (DDL) is primarily affecting the last layers within the model due to the VGE. The DDL model stores its information in those layers more densely, while in PRL, the weight adjustment and information storage are more evenly distributed. The total Shannon entropy is lower in PRL than in DDL.

Table 8. DDL *vs.* PRL comparison at every evolutionary/complexification step.

STEP	DDL Max. score	PRL Mean score	STEP	DDL Max. score	PRL Mean score
0	92.140	**92.144**	10	98.760	**99.161**
1	**98.160**	97.679	11	98.870	**99.173**
2	98.130	**98.347**	12	98.870	**99.179**
3	98.470	**98.879**	13	98.760	**99.193**
4	98.430	**99.007**	14	98.860	**99.186**
5	98.420	**99.048**	15	98.750	**99.192**
6	98.560	**99.115**	16	98.860	**99.239**
7	98.780	**99.105**	17	98.860	**99.239**
8	98.770	**99.135**	18	98.820	**99.243**
9	98.940	**99.138**	19	98.790	**99.258**

Table 8 shows that if at each step we train the same model (resetting its weights first) using DDL, it both achieves lower final accuracy (performs worse) and based on its Shannon entropy score, stores less information. The performance differences between DDL and PRL trained models increases as they become more complex and grow deeper. The Figure 4. DDL and PRL score distribution on Figure 5. Visual graph of experiment 1 shows a visual graph of the results.

3 experimental results are displayed:

- Evolution score and Shannon retrieved during phase 1 of champion 1 creation.
- Mean DDL score and Shannon at each evolutionary step of champion 1 history.
- Mean PRL score and Shannon of the champion model growth by mutation step.

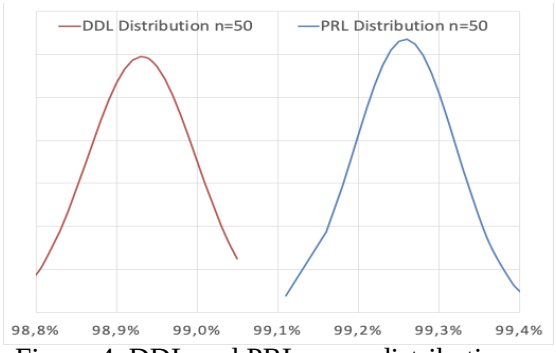

Figure 4. DDL and PRL score distribution on the MNIST dataset.

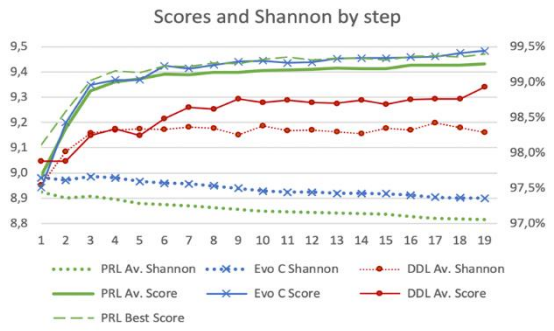

Figure 5. Visual graph of experiment 1 results (MNIST).

Plain lines represent the score, dotted lines represent Shannon entropy and for comparison, the PRL best score is shown as a dashed line. We see that the Shannon score at each step when using DDL of the current step topology is higher (worse) than that of the PRL-based model. Generalization tests (later in this paper) shows that this behavior is reproducible. Further tests must be conducted to conclude whether this behavior applies to any other complex models if we were to build a phylogenetic path and apply the PRL method. Alternatively, perhaps an artificial PRL approach could be used, where any deep model is re-built up one layer at a time and retrained at every step using either an artificially created output layer (of the correct output layer length) until the last layer [29] or by re-attaching the last layer to each consecutive layer and then re-training the model. These artificial approaches of building a path are limited to simple and mostly sequential topologies. Such limitations are not present when it comes to neuroevolution based path building.

5. DISCUSSION

5.1 Reproducibility

This sub-section attempts to answer the following questions: 1/Are these results reproducible with another complex model? 2/ What is the condition of reproducibility? If that condition is the model's complexity, how is such complexity defined?

5.1.1 Reproducibility of PRL Results

To answer the questions, we redo the whole experiment again. For the purpose of reproducibility, we now use another framework, PlaidML and another seed model to generate a new champion. For control, we used the same dataset, the same DDL rules and the same PRL method. The initial model test 2 (Table 9) used in this experiment is narrower, but deeper, as compared to the one in the previous experiment.

Table 9. Initial model test 2.

Layer #	Type	Output	Params
1	InputLayer	N, 28, 28, 1	0
2	Conv2D	N, 27, 27, 6	30
3	MaxPooling2D	N, 9, 9, 6	0
4	Flatten	N, 486	0
5	Dense	N, 10	4870

Table 10 shows the metrics of champion 2 generated from the initial model test 2 (Table 9) during neuroevolution phase 1 of the method.

Table 10. Champion 2 results (MNIST).

Score	Cycle	Generation	Parameters	Nodes	Layers
0.9943	144	28	226 592	39	14

Champion 2 topology generated is smaller, but with a more complex structure, than champion 1 generated in the first experiment. Champion 2 has been generated with 28 evolutionary steps. Furthermore, champion 2 is much harder to train than "initial model 2". Champion 2 epoch time is 15 times that of "initial model 2". Applying DDL to champion 2 gives the following results (MNIST):

DDL average score: 98.90% +/- 0.001 (n=16)
DDL maximum score: 99.08%.

In comparison to the baseline result of the generated champion 2 using neuroevolution, the score we get using DDL with champion 2 topology is lower 99.08% at max. *versus* 99.43% (Table 10). In Table 11, we see that PRL is still more efficient than the DDL approach. The original score of the champion is on average better, which is consistent with our earlier experiments.

Table 11. Results of PRL, DDL applied to champion model 2 (MNIST).

STEP	STD. DEV.	PRL Av. SCORE	DDL Av. SCORE	CHAMP. 2 SCORE	STEP	STD. DEV.	PRL Av SCORE	DDL Av. SCORE	CHAMP. 2 SCORE
0	0.72%	94.31%	**96.37%**	94.60%					
1	0.59%	95.81%	**96.09%**	95.96%	15	0.05%	**99.11%**	98.71%	99.14%
2	0.48%	96.48%	**97.38%**	95.35%	16	0.07%	**99.11%**	98.96%	99.18%
3	0.26%	**97.78%**	97.33%	97.72%	17	0.05%	**99.15%**	98.96%	99.09%
4	0.12%	98.28%	**98.46%**	98.35%	18	0.06%	**99.19%**	98.84%	99.15%
5	0.13%	**98.54%**	98.47%	98.34%	19	0.05%	**99.17%**	98.98%	99.20%
6	0.09%	**98.73%**	98.59%	98.71%	20	0.06%	**99.18%**	98.93%	99.24%
7	0.11%	98.50%	**98.75%**	98.40%	21	0.06%	**99.18%**	98.97%	99.31%
8	0.11%	98.56%	**98.63%**	98.60%	22	0.04%	**99.14%**	98.91%	99.31%
9	0.20%	98.51%	**98.69%**	98.73%	23	0.06%	**99.14%**	98.90%	99.24%
10	0.11%	98.73%	**98.80%**	98.84%	24	0.07%	**99.14%**	99.02%	99.32%
11	0.22%	98.67%	**98.87%**	98.84%	25	0.07%	**99.18%**	98.86%	99.33%

12	0.11%	**98.91%**	98.71%	98.99%	26	0.08%	**99.13%**	98.97%	99.37%
13	0.11%	**98.98%**	98.86%	99.04%	27	0.06%	**99.18%**	98.92%	99.35%
14	0.06%	**99.01%**	98.92%	99.07%	28	0.06%	**99.19%**	98.90%	99.43%

The important result of that experiment is that the initial steps with simpler topology where the VGE is not important had higher scores when using DDL than when using PRL. From step 12 onward, the accuracy/performance achieved by PRL is higher, even though the model was more complex.

5.1.2 Complexity of Model Criteria

When referring to model complexity, we assume that the model has a lot of branches, is deep and is non-sequential. We believe that the more complex (in terms of topology) a model is, the more beneficial it would be to train it using PRL. Thus, in order to further explore these assumptions, we conducted another experiment where we changed the neuroevolutionary phase 1 selection rules.
In this third experiment, we added a rule to the selection process to put more weight on selecting those models which trained the quickest (model training speed was weighted into the final fitness score). With this approach, a model with the same accuracy as another, but with a shorter learning speed (aka epoch time), is selected to enter the HOF. This selection pressure resulted in our system generating champions that are quick to train and less complex, therefore less sensible to vanishing-gradient effect.

Champion 3 generated (MNIST):
Score: 99.40% Shannon: 8.4799

DDL average results for champion 3 model: | PRL average results for champion 3 model:
Score: 99.37% Shannon: 8.4150 | Score: 99.33% Shannon: 8.3535

In this experiment, Shannon value is still lower when using PRL as compared to DDL. But, the difference in the results of this experiment are less drastic.

The PRL complexity definition is therefore not only the topological complexity (total parameters, total nodes and node links), but is also linked to the learning efficiency (amount of time it takes to learn) of the model. The more difficult it is for the model to learn a dataset, the more complex its structure needs to be and the more effect PRL method will have on its training.

5.1.3 Experiment with a Larger Dataset (TinyImageNet)

In this fourth experiment, we used the TinyImageNet dataset with 200 classes and relatively small number of training samples (more difficult to learn) and for reproducibility, no data augmentation. 50 tests were run using DDL and PRL.

Champion 4 generated (TinyImageNet):
19,771,676 parameters, 65 nodes, 30 layers, 44 generations
Score: 42.87% Shannon: 9.3795

DDL average results for champion 3 model: | PRL average results for champion 3 model:
Score : 38.37% Shannon : 9.3833 | Score : 41.79% Shannon : 9.3611

PRL training again produces a better result than DDL.
It should be noted that no data augmentation or extra pre-processing was applied to the dataset. Data augmentation is a common approach with this dataset due to the few samples it contains for each class. Given that our goal is to compare "apples to apples" and find the relative performance of one method compared to another, we applied both PRL and DDL to the original pure TinyImageNet dataset.

5.2 Transferability

In this sub-section, we try to answer the following two questions: 1. Can we use PRL to retrain the model from previous experiments on new datasets from the same problem domain? and 2. Can PRL allow a model to generalize from one dataset to another in the same problem domain better than DDL? In order to answer these questions, we applied PRL to seed and phylogenetic path of champion 1 again, but trained it on a different dataset. The purpose of this experiment is to evaluate whether a model with its recorded evolutionary path from one dataset can be applied on another, but related, dataset.

5.2.1 Experiment 5: FASHION MNIST Dataset Champion 1

The two learning methods are re-applied to the FASHION MNIST dataset. This dataset has the same input and output shape as the standard MNIST. In this dataset, the classification is done on various fashion objects (dresses, shoes, …ext.) rather than digits. This dataset is found to be more complex than the standard MNIST.

DDL mean score for champion 1 - FashionM | PRL mean score for champion 1 - FashionM
Score: 90.44% Shannon: 9.0238 | Score: 91.98% Shannon: 8.4813

This experiment shows that we can re-apply PRL to an existing model and re-train it on a related, but different, dataset. PRL provides a better result than DDL. For comparison, the SOTA convolutional NN applied to the FASHION MNIST is 91.4% without data augmentation [30]. Our 91.98% is a competitive result that outperforms the SOTA, even though the model trained by PRL was not evolved for that specific dataset. This is an interesting result. One potential implication of this result is that PRL might allow us to more easily re-train existing model architectures on new, but related, problem domains for which they were not originally designed and still achieve very high performance.

Table 12. Shannon layer comparison for FASHION MNIST.

LAYER	TYPE	DDL	PRL
2	CONV2D	9.0238	**8.4813**
3	SEPCNV2D	8.307	**8.0439**
4	CONV2D	15.1492	**15.1338**
5	DENSE	14.7804	**14.7331**
6	DENSE	11.6941	**11.6841**
7	CONV2D	14.1506	**13.9888**
8	CONV2D	14.1568	**13.9923**
9	DENSE	12.1115	**12.1017**
10	DENSE	11.6922	**11.6905**
11	DENSE	**17.3703**	17.4783
12	DENSE	**11.6984**	11.71
13	DENSE	**12.3777**	12.5757

Table 12 shows again that PRL is better able to alleviate the VGE. The first layers have better Shannon entropy values when a model is trained through PRL and the last layers have better entropy values when DDL is used to train the model.

5.2.2 Experiment 6: CIFAR10 Gray

In this experiment, we train the model on the CIFAR10 dataset converted into grayscale (C10G). For this experiment, we also scaled it to the 28*28*1 resolution, then gray-scaled it, so that we can use the same model again and test its transferability. This downgraded dataset is much more difficult to train than the MNIST.

DDL mean results for champion 1 - C10G: | PRL mean results for champion 1 - C10G
Score: 54.40 Shannon: 9.1690 | Score: 65.01 Shannon: 8.9345

These results again show the PRL's ability to generalize and retrain an existing model on a new, but related, dataset (with the same shape). In our experiments, PRL is consistently producing better results than DDL, both in terms of accuracy and information density (Shannon entropy values).

5.3 Learning Time

While neuroevolution algorithms are known to need more time to achieve good results (though still significantly less when compared to the time needed by experts to design similar problem specific topologies manually), the PRL adds an alternative to the DDL methods.

Overall, the PRL process requires more steps (19 steps*20 epochs at most in experiment 1), but the Epochs' durations are shorter during the early steps (3 to 15 times less). DDL required at most 60 epochs. Further experiments should be conducted to optimize learning time. We believe that methods like layer-freezing and optimizing the number of epochs will help in this. We conducted preliminary PRL tests

with 4 epochs for half the steps followed by an increase in number of epochs until the last step, where the results were still superior as compared to DDL, while decreasing the needed time.

DDL learning time:	60 epoch*x (x = champion epoch duration in seconds)	= 60.0x
PRL in 1st experiment:	(20 epoch*19 step)*(x/3)	= 126.6x
PRL in 1st experiment with step epoch number optimized:		
	(4 epoch*10 step+12 epoch*9 step)*(x/3)	= 49.3x

PRL takes more time than DDL, but with a simple optimization, the runtime could be reduced below the DDL time.

6. CONCLUSION

Based on our experiments and results, PRL has consistently outperformed DDL, primarily by alleviating the VGE problem. We believe that this is the way in which it functions due to the Shannon entropy values calculated for each layer. These values are lower in deeper layers in the models trained by PRL than those trained by DDL. Furthermore, PRL is more resilient to random-weight initialization as compared to DDL. We re-ran the PRL experiment on the same seed model and with the same phylogenetic path, but with each seed model having randomly generated initial synaptic weights. We found that the performances of the evolved champion models were all very similar and more consistent than when randomly initializing a full model and training it with DDL. Our experiments on transferability also show that the method is effective in retraining models on related datasets. This potentially opens the door to further research into the method's use in transfer learning, where a model with its phylogenetic path can be effectively retrained on another dataset or an updated version of the same dataset. We believe that further research is needed into this domain. The combination of neuroevolution, where model/architecture evolution is synergized with training, will yield better performing systems, as compared to systems where the model is trained all at once (DDL). We think that this method might be particularly effective in training very deep and very complex models, where DDL might struggle. Our future work will concentrate on further expanding and exploring this method.

REFERENCES

[1] D. Silver, David et al., "Mastering the Game of Go with Deep Neural Networks and Tree Search," Nature, vol. 529, pp. 484-489, DOI: 10.1038/nature16961, 2016.

[2] P. Vikhar, "Evolutionary Algorithms: A Critical Review and Its Future Prospects," Proc. of the IEEE Int. Conf. on Global Trends in Signal Process., Inf. Comp. and Comm. pp. 261-265, Jalgaon, India, 2016.

[3] F. Gomez, J. Schmidhuber and R. Miikkulainen, "Accelerated Neural Evolution through Cooperatively Coevolved Synapses," Journal of Machine Learning Research, vol. 9, pp. 937-965, 2008.

[4] R. De Nardi, J. Togelius, O. Holland and S. Lucas, "Evolution of Neural Networks for Helicopter Control: Why Modularity Matters," Proc. of the IEEE Int. Conf. on Evolutionary Computation, pp. 1799-1806, DOI: 10.1109/CEC.2006.1688525, Vancouver, Canada, 2006.

[5] V. Heidrich-Meisner, C. Igel, B. Hoeffding and Bernstein, "Races for Selecting Policies in Evolutionary Direct Policy Search," Proc. of the 26th Annual Int. Conf. on Machine Learning (ICML '09), vol. 51, DOI: 10.1145/1553374.1553426, 2009.

[6] J. Lehman et al., "The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities," Massachusetts Institute of Technology, Artificial Life, vol. 26, no. 2, pp. 274–306, 2020.

[7] F. Such et al., "Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning," arXiv, DOI: 10.48550/arXiv.1712.06567, 2017.

[8] X. Zhang, J. Clune and K. Stanley, "On the Relationship between the OpenAI Evolution Strategy and Stochastic Gradient Descent," arXiv: 1712.06564, DOI: 10.48550/arXiv.1712.06564, 2017.

[9] J. Lehman, J. Chen, J. Clune and K. Stanley, "ES Is More Than Just a Traditional Finite-difference Approximator," Proc. of the Genetic and Evolutionary Computation Conference (GECCO '18), pp. 450-457, DOI: 10.1145/3205455.3205474, 2018.

[10] E. Conti, Edoardo et al., "Improving Exploration in Evolution Strategies for Deep Reinforcement Learning *via* a Population of Novelty-seeking Agents," Proc. of the 32nd Int. Conf. on Neural Information Processing Systems (NIPS'18), pp. 5032–5043, 2017.

[11] J. Metzen, M. Edgington, Y. Kassahun and F. Kirchner, "Performance Evaluation of EANT in the Robocup Keepaway Benchmark," Proc. of the 6th Int. Conf. on Machine Learning and Applications (ICMLA 2007), pp. 342-347, DOI: 10.1109/ICMLA.2007.23, 2008.

[12] F. Gomez, J. Schmidhuber and R. Miikkulainen, "Accelerated Neural Evolution through Cooperatively

Coevolved Synapses," JMLR, vol. 9, pp. 937-965, DOI: 10.1145/1390681.1390712, 2008.

[13] K. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," Evolutionary Computation, vol. 10, pp. 99-127, DOI: 10.1162/106365602320169811, 2002.

[14] E. Real, A. Aggarwal, Y. Huang and Q. Le, "Regularized Evolution for Image Classifier Architecture Search," Proc. of AAAI Conf. on Artificial Intellig., vol. 33, DOI: 10.1609/aaai.v33i01.33014780, 2018.

[15] A. Gaier and D. Ha, "Weight Agnostic Neural Networks," arXiv: 1906.04358, DOI: 10.13140/RG.2.2.16025.88169, 2019.

[16] S. Hochreiter, Untersuchungen zu dynamischen neuronalen Netzen, Diploma Thesis, Josef Hochreiter Institut fur Informatik, Technische Universitat Munchen, Germany, 1991.

[17] F. Informatik, Y. Bengio, P. Frasconi and J. Schmidhuber Jfirgen, "Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies," Chapter of Book: A Field Guide to Dynamical Recurrent Neural Networks, pp. 237 – 243, DOI: 10.1109/9780470544037.ch14, IEEE Press, 2003.

[18] Y. Bengio, P. Simard and P. Frasconi, "Learning Long-term Dependencies with Gradient Descent Is Difficult," IEEE Transactions on Neural Networks, vol. 5, pp. 157-166, DOI: 10.1109/72.279181, 1994.

[19] R. Pascanu, T. Mikolov and Y. Bengio, "On the Difficulty of Training Recurrent Neural Networks," Proc. of the 30th Int. Conf. on Machine Learning, JMLR: W&CP, vol. 28, Atlanta, Georgia, USA, 2013.

[20] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," Proc. of the IEEE Conf. on Comp. Vision and Pattern Recog. (CVPR), pp. 770-778, DOI: 10.1109 CVPR.2016.90, 2016.

[21] X. Glorot, A. Bordes and Y. Bengio, "Deep Sparse Rectifier Neural Networks," Proc. of the 14th Int. Conf. on Artificial Intelligence and Statistics, vol. 15, pp. 315-323, Fort Lauderdale, FL, USA, 2011.

[22] Y. Lecun, L. Bottou, G. Orr and K.-R. Müller, "Efficient BackProp," Chapter in Book: Neural Networks: Tricks of the Trade, vol. 7700, pp. 9-48, DOI: 10.1007\/3-540-49430-8_2, 1998.

[23] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," Journal of Machine Learning Research, vol. 9, pp. 249-256, 2010.

[24] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv: 1502.03167, DOI: 10.48550/arXiv.1502.03167, 2015.

[25] Y. Lecun et al., "Backpropagation Applied to Handwritten Zip Code Recognition," Neural Computation, vol. 1, pp. 541-551, DOI: 10.1162 neco.1989.1.4.541, 1989.

[26] H. Noh, T. You, J. Mun and B. Han, "Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization," Proc. of the 31st Conf. on Neural Inf. Process. Sys. (NIPS), Long Beach, USA, 2017.

[27] S. Enrique, J. Hare and M. Niranjan, "Deep Cascade Learning," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 11, pp. 5475 – 5485, DOI: 10.1109/TNNLS.2018.2805098, 2018.

[28] C. Shannon and W. Weaver, The Mathematical Theory of Communication, Note 78, p. 44, 1963.

[29] J. Schmidhuber, "Learning Complex, Extended Sequences Using the Principle of History Compression," Neural Computation, vol. 4, pp. 234-242, DOI: 10.1162/neco.1992.4.2.234, 1992.

[30] O. Granmo et al., "The Convolutional Tsetlin Machine," arXiv: 1905.09688v5, DOI: 10.48550/arXiv.1905.09688, 2019.

ملخص البحث:

رغم التّطور الكبير الذي طرأ على تدريب الشبكات العصبية، إلّا أنّه ولسوء الحظّ، فإنّ نقطة القوة في الشبكات العصبية العميقة – وهي عمقها - تشكل مشكلةً وهي تضاؤل الميْل؛ وذلك لصعوبة التّدريب المعمَّق للطّبقات الأعمق.

تقترح هذه الورقة طريقة "تعلُّم الإعادة المكتسب خلال التّطوّر النوعي"، وهي طريقة تعلُّم تتغلّب بشكلٍ جوهري على مشكلة تضاؤل الميْل. وعلى العكس من طرق التعلُّم بالبواقي، فإنّ الطريقة المقترحة لا تُقيّد بِنية النّموذج. وبدلاً من ذلك، فهي تقوّي عناصر من التّطوُر العصبي، وتعلُّم النّقل، والتّدريب طبقةً طبقةً. ونبين في هذه الورقة أنّ الطريقة المقترحة الجديدة قادرة على إنتاج نموذجٍ ذي أداءٍ أفضل. وبحساب إنتروبيا شانون للأوزان، نبين أنّ الطبقات الأعمق يتمّ تدريبها على نحوٍ أكثر تفصيلاً بحيث تحتوي على معلومات أكثر بدلالةٍ إحصائية مقارنةً بالحالات التي يتمّ فيها تدريب النّموذج بالطّرق التّقليدية.