Augmenting Novelty Search with a Surrogate Model to Engineer Meta-Diversity in Ensembles of Classifiers

Rui P. Cardoso[1], Emma Hart[4], David Burth Kurka[2], and Jeremy Pitt[3]

[1] Imperial College London, rui.cardoso@imperial.ac.uk
[2] d.kurka@imperial.ac.uk
[3] j.pitt@imperial.ac.uk
[4] Edinburgh Napier University, e.hart@napier.ac.uk

Abstract. Using Neuroevolution combined with Novelty Search to promote behavioural diversity is capable of constructing high-performing ensembles for classification. However, using gradient descent to train evolved architectures during the search can be computationally prohibitive. Here we propose a method to overcome this limitation by using a *surrogate* model which estimates the behavioural distance between two neural network architectures required to calculate the sparseness term in Novelty Search. We demonstrate a speedup of 10 times over previous work and significantly improve on previous reported results on three benchmark datasets from Computer Vision — CIFAR-10, CIFAR-100, and SVHN. This results from the expanded architecture search space facilitated by using a surrogate. Our method represents an improved paradigm for implementing horizontal scaling of learning algorithms by making an explicit search for diversity considerably more tractable *for the same bounded resources*.

Keywords: diversity · ensemble · novelty search · surrogate

1 Introduction

Ensemble performance is fundamentally dependent on both the accuracy of individual base learners and the diversity between them [4]. However, techniques to promote diversity are typically only implicit, such as training the models on different subsets of the data or starting from different random initialisations. In previous work [2], we proposed a method that *explicitly* searched for behavioural diversity amongst a set of base learners. This used a Novelty Search (NS) algorithm in conjunction with Neuroevolution, in which novelty was determined by novel metrics that explicitly measured behavioural diversity. The evolved behaviourally diverse ensembles outperformed both their individual learners and ensembles created with techniques that only implicitly promote diversity, and enabled us to study and compare different definitions of diversity.

However, a fundamental limitation of this approach was its computational complexity. In order to calculate the behavioural distance between two models,

we need to compare the classification errors that they make on a validation data set. This requires first training each member of the current population of neural network models on a training data set with gradient descent at each iteration of the NS. If computational resources are limited, this very time-consuming step can be prohibitive. This poses significant challenges because it restricts the search to only a few iterations at best and renders the problem intractable at worst. Such time and computational demands compromise the goal of our approach, which is to develop learning algorithms which scale horizontally, namely with models which can be distributed across many low-cost machines. The claims to tackling the unwieldiness of Deep Learning (DL) algorithms with more scalable solutions were undermined by the fact that our approach was so computationally intensive.

In order to overcome the costly step of training each model, we have introduced a surrogate model [15] into our method. We use this surrogate model, pretrained on a sample drawn from the search space of neural network architectures, to get an estimate of the *error* distance between two neural networks given architectural descriptors, *without* training these networks. Whereas this calculation had previously been a very costly step, this technique renders it essentially instantaneous. This produces a speedup of 10 times compared to the previous approach when the same parameters are used, *without loss of performance*. By changing the parameters to expand the search space of neural network architectures we have considerably improved on previous results reported on three benchmark datasets from Computer Vision — CIFAR-10, CIFAR-100, and SVHN. Using a surrogate model has enabled us to search a wider space of neural network architectures and run the Novelty Search procedure for longer.

The major contribution of this paper is that it proposes an improved paradigm for implementing horizontal scaling of learning algorithms. Explicitly creating diversity amongst the members of an ensemble establishes a sound criterion for distributing these models. By improving the method with a surrogate model in the way described above, our approach makes an explicit search for diversity considerably more tractable *for the same bounded resources*.

2 Background

Combining an evolutionary algorithm (EA) with a surrogate modelling function has been common in the literature for many years, e.g. in single-objective optimisation [19], multi-objective optimisation [14], and particularly in expensive optimisation [23]. A first surrogate model for neural network optimisation was introduced by Gaier *et al.* [5] and used in conjunction with the NEAT [16] algorithm for evolving the weights and topology of a neural network. This paper used a surrogate distance-based model, employing a genotypic compatibility distance metric that is part of NEAT. The approach has been quickly adopted in the literature using a range of surrogates and a variety of methods to evolve networks. There are several examples of approaches that use surrogates to estimate the performance of an architecture. For example, in 2017 Deng *et al.* proposed

the Peephole algorithm [3], which predicted the performance of a convolutional neural network based on its architecture information: a long-short term memory (LSTM) neural network was used to train the model. Stork *et al.* [17] extended a Cartesian Genetic Programming method called CPGANN to evolve neural networks using surrogate-based optimisation to reduce the number of fitness evaluations required. They used a Kriging model [9] as the surrogate. In [18], a Random Forest algorithm (RF) was used as a surrogate to predict the performance of a CNN architecture — the authors proposed a method for describing a CNN as a set of features which were used as input to the RF. In [15], the authors use a surrogate benchmark for neural architecture search (NAS).

In contrast, Hagg *et al.* [7] introduce a more flexible method for building a surrogate model that is independent of network topology: rather than describing the neural network architecture, they introduce a *phenotypic* metric which measures the difference in output between two neural networks given the same input sequence. The difference is used in a Kriging surrogate model. Our proposed approach is conceptually closest to that of Hagg. For a given neural network, we calculate a behavioural vector that describes its behaviour on a dataset (see Section 3.2). We then propose a RF surrogate model that is used to estimate the distance between the behavioural vectors produced by any two neural networks, as this value is required to drive a NS algorithm.

3 Methods and Materials

We use NS to evolve an ensemble of behaviourally diverse neural network models. The NS operates over a space of architectures defined by a set of hyperparameters. Unlike our previous work [2], where the neural networks in a generation had to be trained with gradient descent at each iteration of the NS in order to calculate the behavioural distances between each pair of architectures, these distances are now *estimated* by a surrogate model which is pretrained on a sample drawn from the space of neural network architectures. The most diverse models are added to the final ensemble, which is then trained on the input data. We evaluate the method against our previous method and compare the performance obtained with different diversity metrics. The following subsections go into detail about each of these steps.

3.1 Neural Network Architectures

The architectures evolved by our procedure are *residual neural networks* [8] based on the wide architectures proposed by [22]. They are of the same kind as those we used in our previous work. Figure 1a shows a generic neural network and Figure 1b illustrates a generic residual block. Please refer to our previous paper [2] for a more detailed description of these architectures.

The *hyperparameters* of each network are evolved by NS. Each individual in the population is defined by a variable-length vector, depending on the number of blocks r: $[J, C, O^1, ..., O^r, D^1, ..., D^r]$, where J is a Boolean value indicating

whether the network should be trained jointly or separately if it is in the final ensemble, C is the output size of the first convolution, O^i is the output size of block i, and D^i its dropout probability. Each individual is mapped to a Pytorch module [13] for implementation purposes. The *parameters* of each network are randomly initialised and then optimised by a standard gradient descent procedure.

In order to preprocess the input to the surrogate model, we *normalise* the representation described above in the following way: we first *rescale* the elements in all positions so that they lie between 0 and 1. The first element is the Boolean value indicating whether the neural network should be trained jointly or separately, so it need not be normalised. Then, given that the representations have variable length depending on the number of residual blocks in each neural network, we *pad* the vector so that it has fixed length, corresponding to the maximum possible number of residual blocks, by adding an appropriate number of elements equal to 0 before the sequence of block output sizes and before the sequence of dropout probabilities. Therefore, if the number of residual blocks in the network is r and the maximum number of blocks is R; if the maximum and minimum sizes of the first convolution in the network are C_{max} and C_{min}, respectively; if the maximum and minimum sizes of each residual block are O_{max} and O_{min}, respectively; and if the maximum and minimum dropout probability of each block are D_{max} and D_{min}; then the normalised representation of neural network m_i is:

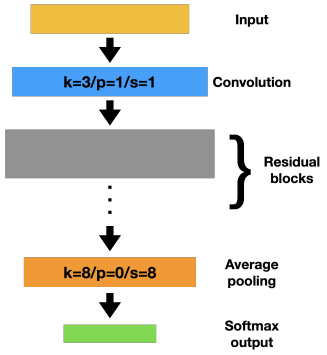
$$\begin{aligned}\mathbf{norm\_rep}_i = \Bigg[& J_i, \frac{C_i - C_{min}}{C_{max} - C_{min}}, \\ & 0, ..., 0, \frac{O_i^{R-r} - O_{min}}{O_{max} - O_{min}}, ..., \frac{O_i^R - O_{min}}{O_{max} - O_{min}}, \\ & 0, ..., 0, \frac{D_i^{R-r} - D_{min}}{D_{max} - D_{min}}, ..., \frac{D_i^R - D_{min}}{D_{max} - D_{min}} \Bigg]\end{aligned} \tag{1}$$

Where there are $R-r$ elements equal to 0 before the sequence of block output sizes and before the sequence of dropout probabilities.

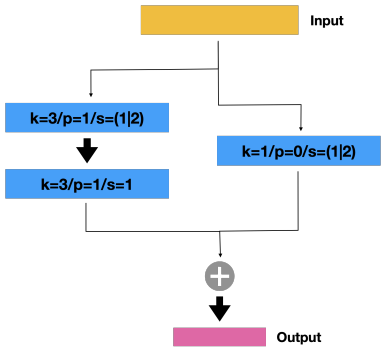
3.2 Diversity Metrics

In order to calculate novelty scores, which are used as the objective function by the NS, we have considered six different diversity metrics, five of which we have defined ourselves. These metrics are calculated between each pair of individual neural network architectures. We have used three of these metrics in the previous version of our procedure [2].

Let $\boldsymbol{y}_i$ be the vector of predictions for model m_i with each prediction y_i^n for data point $\boldsymbol{x}^n$ being a class label in $\{1..C\}$. Let $\boldsymbol{p}_i$ be a binary vector where $p_i^n = 1$ if the prediction y_i^n is correct and $p_i^n = 0$ otherwise. Let N^{11}, N^{00}, N^{01}, and N^{10}, respectively, be the total number of test instances where two models are both correct, both incorrect, and when one is correct and the other is not. The



(a) Generic residual neural network (k = kernel size; p = padding; s = stride)



(b) Generic residual block

Fig. 1: Generic topology of individual neural networks

first diversity metric we consider is the *proportion of different errors* between two models when at least one of them is *correct*. We propose this metric since it provides insight into the divergence between the errors made by two models. We have defined it as:

$$\text{prop}^1_{i,j} = \frac{N^{01} + N^{10}}{N^{11} + N^{01} + N^{10}} \tag{2}$$

The second diversity metric we consider is very similar and is the *proportion of different errors* between two models when at least one of them is *incorrect*. We have first proposed this metric in our previous work [2], defining it as:

$$\text{prop}^2_{i,j} = \frac{N^{01} + N^{10}}{N^{00} + N^{01} + N^{10}} \tag{3}$$

The third metric we propose is the *harmonic mean* between these two proportion metrics. This is a sound way of averaging the two proportion metrics into a single metric so that they are both taken into account. It is defined as:

$$\text{prop}^{\text{harm}}_{i,j} = \frac{2 \cdot \text{prop}^1_{i,j} \cdot \text{prop}^2_{i,j}}{\text{prop}^1_{i,j} + \text{prop}^2_{i,j}} \tag{4}$$

We also consider a widely used metric (e.g. [20, 10, 12]) defined as the *disagreement* between two models, i.e. the proportion of test instances where one of them is correct and the other is not. We take this metric into account since it expresses how commonly two models disagree on any test instance. It is defined as:

$$\text{dis}_{i,j} = \frac{N^{01} + N^{10}}{N^{00} + N^{01} + N^{10} + N^{11}} \tag{5}$$

Consider now the *two's complement* of the binary vector of correct predictions $\boldsymbol{p}_i$, $\boldsymbol{w}_i$, i.e. the binary vector of *wrong* predictions. The next metric we propose

is the *cosine distance* between the binary vectors of wrong predictions made by two models m_i and m_j. Like $\text{prop}^1_{i,j}$ and $\text{prop}^2_{i,j}$, we consider this metric because it is a measure of the distance between the errors made by two models. We have defined it as:

$$\text{cos\_dist}_{i,j} = 1 - \frac{\boldsymbol{w}_i \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\|\|\boldsymbol{w}_j\|} \tag{6}$$

At last we consider a metric of *architectural* diversity. Take the *normalised* vector which represents each individual neural network, as described in section 3.1. Let its size be L. To obtain an architectural representation, we simply remove the first element from the normalised representation, i.e. the Boolean value indicating whether or not the neural network should be trained separately or jointly. Thus, referring to Equation 1, the architectural representation of model m_i is:

$$\textbf{arch\_rep}_i = \textbf{norm\_rep}_i^{\{1..L-1\}} \tag{7}$$

We then define *architectural distance* between neural networks m_i and m_j as the cosine distance between their normalised architectural representations:

$$\text{arch\_dist}_{i,j} = 1 - \frac{\textbf{arch\_rep}_i \cdot \textbf{arch\_rep}_j}{\|\textbf{arch\_rep}_i\|\|\textbf{arch\_rep}_j\|} \tag{8}$$

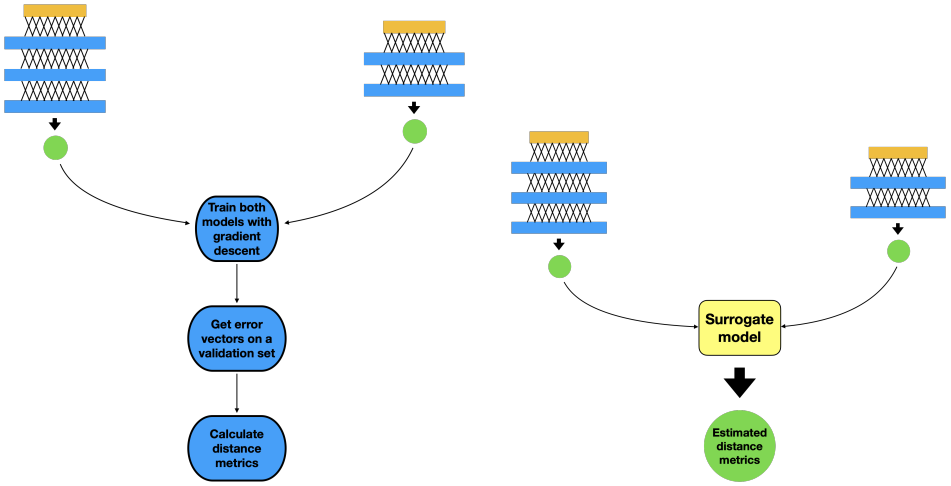
These metrics determine the *behavioural distance* between two neural network models, which is used to calculate the novelty scores that guide the NS procedure, as explained in Section 3.5. Note that the metrics $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$ focus more closely on the instances where the models made a *prediction error*. In our previous work [2], these two metrics have led to better performance than the others. Here we are interested in learning whether the same pattern can be observed with our new improved version of the NS procedure.

3.3 Surrogate Model to Estimate Distances

The NS requires novelty scores to be determined, which in turn require the distances between pairs of neural networks in the current population to be calculated. However, calculating the exact distance values between two neural network models entails first training the models on the input data with gradient descent and then evaluating them on a validation dataset, as we did in previous work [2]. This can be a very costly step if computational resources are limited, which constrains the NS to only a few iterations and the population to a small size — as the neural networks have to be trained in parallel for efficiency. Here we overcome this limitation by pretraining a Random Forest [1] surrogate model which estimates the behavioural distances between a pair of neural network models.

Note that the estimates of behavioural distances produced by the surrogate model do not need to be very accurate. This is because, when calculating the novelty score of a particular individual neural network, we only need to know relative distances in order to determine nearest neighbours. This means that the surrogate model need only capture the general trends of growth of the distance

values, even if the actual values are not very precise. This makes the use of a surrogate model very appropriate with no need for a very complex model. Figure 2 shows the differences between the previous method for calculating exact distance values, shown in Figure 2a, and the current method using a surrogate model, shown in Figure 2b. Calculating exact distance values is a very costly step, potentially requiring several GPU hours depending on the length of training. In contrast, estimating these distances by means of the surrogate model is an instantaneous process, once the surrogate model has been trained on sample data beforehand.



(a) Steps required to calculate exact distance values between two neural networks

(b) Estimating the distance values between two neural networks with a surrogate model

Fig. 2: Difference between calculating and estimating distance values

3.4 Pretraining the Surrogate Model

The surrogate model must be trained beforehand so that it can be used effectively during the NS to estimate the distance values between two neural network models. To do this, we draw a sample of neural networks from the search space of architectures defined by the set of hyperparameters used with the NS method. We first train each of these neural networks with gradient descent and calculate their error vectors on a validation data set. We then build random pairs of neural networks and calculate the exact distance values, for all six metrics considered, between them as a function of either their error vectors or their architectural descriptors, as explained in Section 3.2. Finally, we construct a data set on which we fit a Random Forest regressor [1] which takes as input the normalised representations of two neural network architectures, as per Section 3.1, and has six outputs: the estimates of the distance values for all six metrics considered. We

Algorithm 1 Pretraining the surrogate model on sample architectures

draw a sample S of neural networks from the search space defined by J, $[C_{min}, C_{max}]$, $[O_{min}, O_{max}]$, and $[D_{min}, D_{max}]$;
train(S); ▷ Models trained jointly or separately according to the value of J_i
for neural network model $m_i \in S$ **do**
  get error vector e_i on validation set $\mathcal{D}_{val}$;
end for
build $\frac{\|S\|^2 - \|S\|}{2}$ unique pairs of neural networks;
initialise dataset $\mathcal{D}_{dists} \leftarrow \emptyset$;
for each pair m_i, m_j **do**
  $\boldsymbol{d} \leftarrow$ all 6 distance values; ▷ calculated as per Section 3.2
  norm_rep_i, norm_rep_j are the normalised representations of m_i and m_j;
  add data point $x \leftarrow \left\{\text{norm\_rep}_i, \text{norm\_rep}_j, \boldsymbol{d}\right\}$ to $\mathcal{D}_{dists}$;
end for
train random forest model rf on $\mathcal{D}_{dists}$;
return random forest model rf;

have selected a Random Forest model due to its low complexity and because we expect it to generalise well on new data, given that it is an ensemble model. Algorithm 1 describes the process of training this surrogate model in pseudocode.

3.5 Novelty Search Algorithm

Our algorithm for building an ensemble implements NS as described by [11], applying it to our problem domain. Algorithm 2 presents the pseudocode for this procedure. The original training data is split into two sets, one for training and one for validation. The training set is used to train the final ensemble; it is also used to train the sample of neural network architectures drawn from the search space that is in turn used to pretrain the surrogate model. Whereas training each of these sample neural networks makes use of the entire training set, pretraining the surrogate model only requires the validation set, which is used to calculate exact distance values between pairs of neural networks.

Selection in NS is driven by the novelty score, which computes the sparseness at any point in the behavioural space. This sparseness is defined by one of the distance metrics of Section 3.2. Areas with denser clusters of visited points are considered less novel and therefore rewarded less. This is defined as the average distance to the K-nearest neighbours of a point, calculated with respect to the other individuals in the current generation and to a stored *archive* of previously sampled solutions. Hence, the novelty score is calculated as:

$$NS_i = \frac{1}{k} \sum_{k=0}^{K} \text{div\_metric}(m_i, \mu_k) \tag{9}$$

Where μ_k is the kth-nearest neighbour of m_i with respect to the diversity metric $\text{div\_metric}_{i,j}$, selected from the metrics defined in Section 3.2.

Individuals are selected for reproduction on the basis of their novelty scores using a tournament selection procedure. In the interests of promoting divergence

Algorithm 2 Ensemble evolution through NS

randomly initialise population *pop*;
$archive \leftarrow \emptyset$;
$elite\_archive \leftarrow \emptyset$;
draw $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ from training set $\mathcal{D}$;
set evolution iterations *epochs*;
set final ensemble size *ensemble_size*;
surrogate model rf pretrained as per Section 3.3;
select diversity div_metric$_{i,j}$ from Section 3.2;
ns_i is the fitness value (novelty score) of model m_i;
for *epochs* **do**
  use surrogate model to estimate the distance between each pair of neural network models in the population and between each model in the population and all models in the archive;
  calculate novelty score of every member of the population as per Equation 9;
  in addition to the novelty score, calculate an *elite score* for each model by first estimating the distance between each model in the population and all models in the elite archive;
  add a random sample of the population to the archive;
  add individual with highest elite score to the elite archive;
  select models for reproduction using tournament selection;
  generate a new population by applying the mutation operator on the selected models;
end for
calculate distance values between models in the elite archive;
calculate ensemble scores based on these distance values;
final ensemble is the top *ensemble_size* models with highest ensemble score;
train(*ensemble*); ▷ Models trained jointly or separately according to the value of J_i

and avoiding convergence, reproduction only uses mutation. Mutation either adds or removes a randomly chosen residual block from an individual, modifying input/output sizes at the mutation point as necessary; changes the output size and dropout probability of a random block; or swaps two consecutive blocks chosen at random.

After evaluating the entire population, n_A randomly chosen individuals are added to the *archive*, following the method suggested in [6]. In addition, the individual from the population with the highest *elite score*, calculated in a similar fashion to the novelty score, is added to an *elite archive*. After running the NS for the specified number of iterations, a subset of this elite archive is selected as the final ensemble. This subset is chosen so as to maximise the average distance amongst its members. The final ensemble is then trained by gradient descent, the *only time* when this parameter optimisation takes place.

3.6 Evaluation of Evolved Ensembles

In order to evaluate the performance of the evolved ensemble, we use the stacking technique [21], which trains a linear model to weight the predictions of each

individual learner. This linear model is trained for a configurable number of iterations on the validation set mentioned in Section 3.5. This is to avoid overfitting the test set.

3.7 Baseline: Previous Method

The approach we present here is an improvement of the method that we first proposed in [2]. We use this as a baseline against which we compare the new method. In its main aspects, the previous method is similar to the new method, with the notable difference that it does not make use of a surrogate model to estimate the distance between two neural network models. As discussed previously, this original method calculates exact distances between each pair of neural networks by first training all the models in the current generation with gradient descent and then getting their error vectors by evaluating them on a validation set. As an additional difference, the previous method would calculate at each iteration an *ensemble selection metric* for each member of the population and then add to the final ensemble the single best-scoring neural network in each generation. The new method maintains an *elite archive*, to which a sample of neural networks from each generation with highest novelty score with respect to this archive, which we call *elite score*, is added at each iteration; novelty scores with respect to the final elite archive are calculated at the end for each of its members and this *ensemble score* is used to select a subset of neural networks which will make up the final ensemble.

We compare the new method proposed in this paper with our previous approach along two main lines. Firstly, we seek to understand whether there is a speedup with the new method as a result of increased efficiency when looking for solutions of similar complexity to those found with the previous method. Secondly, we investigate whether the new method can be used to produce better solutions, i.e. solutions of higher complexity and leading to better final performance. This means that we are interested in investigating whether there is both a quantitative improvement, i.e. being able to do *more of* what could be done with the previous method thanks to a more efficient use of computational resources, and a qualitative improvement, i.e. being able to do *more than* what could be done with the previous method by tackling solutions that were previously unfeasible or intractable.

4 Experiments

This section describes the two sets of experiments carried out for comparing the new NS method, which makes use of a surrogate model to estimate the distance between models as described previously, with the previous method, which instead calculates exact distance values by first training all the models in the population with gradient descent and then determining their error vectors on a validation set. We compare the methods based on resource usage, namely runtime, for similar parameter settings and model complexity, as well as on their ability to scale to larger search spaces and search for more complex models.

4.1 Test set 1: Resource Usage for Similar Complexity

In this set of tests, we investigate the total time required to run each of the two methods when they are looking for *solutions of the same complexity* and running for the same number of iterations. We wish to determine the speedup that can be gained with the new method, which makes use of a surrogate model to overcome the need for training all the models in the current generation with gradient descent in order to calculate novelty scores. We run both the new and the previous methods on CIFAR-10 and fix the parameters, as shown in the second column of Table 1. We conjecture that in these conditions our new method not only results in a speedup due to the use of a Random Forest surrogate model, but also outputs ensembles of similar performance. This is expressed by Hypotheses 1 and 2.

Hypothesis 1 (Runtime of previous NS method vs new method enhanced with a surrogate model) *Enhancing the NS procedure with a Random Forest surrogate model pretrained to estimate the distance between models, and thereby their novelty scores, results in a speedup compared with our previous method, which calculates exact distance values and novelty scores, when constructing ensembles of the same complexity.*

Hypothesis 2 (Performance achieved with the previous NS method vs the new method with a surrogate model) *When looking for solutions of the same complexity, the new NS procedure, using a surrogate model, outputs ensembles which do not perform worse than those constructed by our previous method, even though the new method only* estimates *distance values and novelty scores.*

4.2 Test set 2: Expanding the Search Space

Using a surrogate model to speed up the procedure has enabled us to both search for solutions of higher complexity and run the NS for longer. In this set of experiments, we apply the new method to three benchmark datasets from the Computer Vision (CV) literature — CIFAR-10, CIFAR-100, and SVHN — and test it with all diversity metrics previously defined in Section 3.2. We also compare the results achieved with the new method to the best results observed with the previous method. The parameters that we use with the new method are shown in the third column of Table 1. We expect to see further evidence of what we observed in previous work [2] regarding *error diversity* metrics, namely that those diversity metrics which focus more closely on the instances where the models make prediction errors lead to higher-performing ensembles. This is expressed by Hypothesis 3. We also expect the new method to lead to higher-performing ensembles than those constructed with the previous method, since the use of a surrogate model makes it feasible to expand the search space and run the NS for longer. This is expressed by Hypothesis 4.

Table 1: Novelty search parameters for both test sets

Parameter	**Test set 1: runtime comparison (both methods)**	**Test set 2: expanded search space (new method only)**
Iterations	10	100
Final ensemble size	11	40
Population size	30	100
Diversity metric	$\cos\_dist_{i,j}$	All from Section 3.2
Number of blocks	2:6	2:6
Number of channels in the first convolution	4:16	4:16
Number of channels in residual blocks	24:32	16:64
Dropout probability in residual blocks	0.1:0.4	0.1:0.9
Number of neighbours K	3	15
Size n_A of archive sample	5	10
Size of tournament for selection	10	50

Hypothesis 3 (Better performance with metrics that focus on error instances) *In a similar fashion to what we have observed with our previous method, running the NS procedure with the distance metrics that focus more closely on the instances where the models make prediction errors leads to higher-performing ensembles than when more generic diversity metrics are employed.*

Hypothesis 4 (Performance achieved with the previous NS method vs the new method with a surrogate model) *The new NS method enhanced with a surrogate model makes it possible to search a larger space of more complex neural network architectures and, therefore, outputs higher-performing ensembles than the best ones constructed by our previous method.*

5 Results and Discussion

In this section, we present the results of the two sets of experiments described in Section 4. We then discuss these results and whether the hypotheses formulated above can be rejected.

5.1 Hypothesis 1

Table 2 shows the median value, calculated after 10 independent runs, of the time required to run both the previous NS method and the new method, which makes use of a surrogate model, with the same parameters. These results show that the new method is about 10 times faster than the original NS method. A Mann-Whitney significance test shows that this difference is significant at

Table 2: Median results over 10 runs of the previous NS method and the new NS method with a surrogate model on CIFAR-10 (test set 1)

Runtime of NS	48760.5 s
Runtime of NS with surrogate model	4871 s
Training a sample of architectures	28970.5 s
Building a dataset and training the Random Forest surrogate model	18113.5 s
Accuracy achieved by NS	82.245%
Accuracy achieved by NS with surrogate model	83.885%

the 1% level. This supports the claim of Hypothesis 1 that enhancing the NS method with a Random Forest surrogate model to estimate the distances between models speeds up the search for diverse models and the construction of a diverse ensemble. For reference, we also report in Table 2 the median time, over 10 runs, required to train a sample of 40 neural network architectures on CIFAR-10, as well as to build a dataset and train the Random Forest surrogate model as per Algorithm 1. Note that these two runtimes are a *one-off cost* and that, in order to pretrain the surrogate model for our experiments, we have trained a total of 3200 sample architectures by running several processes in parallel on a cluster, each training 40 architectures.

5.2 Hypothesis 2

Table 2 also shows the median accuracy, calculated after 10 independent runs, achieved by ensembles constructed by both the previous NS method and the new method, when these are executed with the same parameters. The results show that the ensembles constructed by the new method do not perform worse than those constructed by the original method, which calculates exact values for the distance metrics and novelty scores. In fact, we observe that the new method leads to slightly better performance. A Mann-Whitney significance test shows that this difference is significant at the 1% level. This corroborates Hypothesis 2, which claims that there is no loss in performance when using the new method and its surrogate estimates. Besides the use of surrogate models, the major difference between the previous and the new method is the way a subset of all the models is selected to be in the final ensemble. As explained before, the previous method applies an *ensemble selection metric* at each iteration of the NS, whereas the new method keeps an *elite archive*, from which the final ensemble is selected in an additional step at the end of the procedure. It seems that the ensemble selection procedure of the new method is the cause behind the better performance achieved by its ensembles.

Table 3: Median accuracy over 10 runs of ensembles constructed by the new method (test set 2). Best results highlighted. Best results with the original NS shown for comparison

Dataset	Diversity metric	Final ensemble accuracy	Best accuracy with original NS (from [2])
CIFAR-10	$\text{prop}^1_{i,j}$	67.295%	83.51%
	$\text{prop}^2_{i,j}$	90.605%	
	$\text{prop}^{\text{harm}}_{i,j}$	83.975%	
	$\text{dis}_{i,j}$	86.28%	
	$\text{cos\_dist}_{i,j}$	90.11%	
	$\text{arch\_dist}_{i,j}$	80.4%	
CIFAR-100	$\text{prop}^1_{i,j}$	28.725%	45.42%
	$\text{prop}^2_{i,j}$	63.05%	
	$\text{prop}^{\text{harm}}_{i,j}$	63.41%	
	$\text{dis}_{i,j}$	63.18%	
	$\text{cos\_dist}_{i,j}$	63.035%	
	$\text{arch\_dist}_{i,j}$	49.83%	
SVHN	$\text{prop}^1_{i,j}$	78.825%	91.435%
	$\text{prop}^2_{i,j}$	94.8%	
	$\text{prop}^{\text{harm}}_{i,j}$	89.775%	
	$\text{dis}_{i,j}$	90.675%	
	$\text{cos\_dist}_{i,j}$	94.79%	
	$\text{arch\_dist}_{i,j}$	90.68%	

5.3 Hypothesis 3

Table 3 shows the median accuracy, after 10 runs, of ensembles evolved by the new NS procedure extended with a surrogate model, for all six diversity metrics of Section 3.2 and all three datasets considered. We observe that on CIFAR-10 and SVHN, the metrics $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$ lead to the highest-performing ensembles. Mann-Whitney tests show that the difference to the other metrics is statistically significant. On CIFAR-100, this is observed additionally with the metrics $\text{prop}^{\text{harm}}_{i,j}$ and $\text{dis}_{i,j}$.

The metrics $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$ are the two that focus more closely on the instances where the two models being compared make prediction errors. Additionally, the metric $\text{prop}^{\text{harm}}_{i,j}$ depends on the value of $\text{prop}^2_{i,j}$. These observations back the claim of Hypothesis 3 that error diversity metrics lead to better-performing ensembles compared to more generic diversity metrics. This confirms what we observed in our previous work [2].

5.4 Hypothesis 4

The last column of Table 3 shows the best performance achieved by ensembles evolved with our previous NS method. These results show very clearly that the

new method constructs higher-performing ensembles than our previous procedure, with the most considerable difference being observed on CIFAR-100 and CIFAR-10. Mann-Whitney tests reveal that, for each dataset, the difference between the best results achieved by the new method and the best achieved by the previous method is indeed statistically significant. This difference results from the fact that the new method, thanks to its use of a surrogate model, is able to *search a wider space of neural network architectures*, even though it runs on *the same bounded resources*. We conclude that this supports Hypothesis 4.

6 Conclusions and Future Work

This paper has extended on previous work [2], which proposed an innovative NS method to build behaviourally diverse ensembles of classifiers. The previous method had signposted an innovative way to construct high-performing ensembles by explicitly searching for diversity. However, its application in practice had been hampered by limitations in the amount of available computational resources, since it involved a time-consuming step of training all networks in each generation of the NS with gradient descent. Our new method overcomes this limitation by using a pretrained surrogate model to estimate the distance between neural network architectures, necessary to calculate novelty scores, without the need to train them. In this way, we can obtain an approximate speedup of 10 times w.r.t. the previous method when running them both with the same parameters, *without loss of classification accuracy*. We can also construct better-performing ensembles thanks to the expanded architecture search space facilitated by using a surrogate. We have confirmed previous observations that error diversity metrics lead to better-performing ensembles than more generic metrics.

Our method thus represents an improved paradigm for implementing horizontal scaling of learning algorithms. It makes an explicit search for diversity considerably more tractable than our original approach *for the same bounded resources*. In future work, we will extend the current method by implementing a local competition (LC) variant, so that it is possible to include objectives of accuracy into the NS. This will enable us to further study the relationship between diversity and classification accuracy and to investigate trade-offs between the two. We will also propose more definitions of diversity with new and improved metrics, which will provide more insight into what makes a good diversity metric that fits the task of constructing a diverse high-performing ensemble.

References

1. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
2. Cardoso, R.P., Hart, E., Kurka, D.B., Pitt, J.V.: Using novelty search to explicitly create diversity in ensembles of classifiers. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 849–857. GECCO '21, Association for Computing Machinery, New York, NY, USA (2021)
3. Deng, B., Yan, J., Lin, D.: Peephole: Predicting network performance before training. arXiv preprint arXiv:1712.03351 (2017)

4. Dietterich, T.G.: Ensemble methods in machine learning. In: International workshop on multiple classifier systems. pp. 1–15. Springer (2000)
5. Gaier, A., Asteroth, A., Mouret, J.B.: Data-efficient neuroevolution with kernel-based surrogate models. In: Proceedings of the genetic and evolutionary computation conference. pp. 85–92 (2018)
6. Gomes, J., Mariano, P., Christensen, A.L.: Devising effective novelty search algorithms: A comprehensive empirical study. In: GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference (2015)
7. Hagg, A., Zaefferer, M., Stork, J., Gaier, A.: Prediction of neural network performance by phenotypic modeling. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 1576–1582. GECCO '19, Association for Computing Machinery, New York, NY, USA (2019)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
9. Jean-Paul, N.C., Desassis: Fifty years of kriging (2018). https://doi.org/10.1007/978-3-319-78999-6_29, https://doi.org/10.1007/978-3-319-78999-6_29
10. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning (2003)
11. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary Computation (2011)
12. Pasti, R., De Castro, L.N., Coelho, G.P., Von Zuben, F.J.: Neural network ensembles: Immune-inspired approaches to the diversity of components. Natural Computing **9**(3), 625–653 (2010)
13. Paszke, A., Gross, S., Chintala, S., et al.: Automatic differentiation in PyTorch. In: Advances in Neural Information Processing Systems 32 (2019)
14. Ruan, X., Li, K., Derbel, B., Liefooghe, A.: Surrogate assisted evolutionary algorithm for medium scale multi-objective optimisation problems. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. pp. 560–568 (2020)
15. Siems, J., Zimmer, L., Zela, A., et al.: NAS-Bench-301 and the case for surrogate benchmarks for neural architecture search (2020)
16. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
17. Stork, J., Zaefferer, M., Bartz-Beielstein, T.: Improving neuroevolution efficiency by surrogate model-based optimization with phenotypic distance kernels. In: International Conference on the Applications of Evolutionary Computation (Part of EvoStar). pp. 504–519. Springer (2019)
18. Sun, Y., Wang, H., Xue, B., et al.: Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. IEEE Transactions on Evolutionary Computation (2019)
19. Tong, H., Huang, C., Minku, L.L., Yao, X.: Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study. Information Sciences **562**, 414–437 (2021)
20. Van Krevelen, R.: Error Diversity in Classification Ensembles. Ph.D. thesis (2005)
21. Wolpert, D.H.: Stacked generalization. Neural Networks (1992)
22. Zagoruyko, S., Komodakis, N.: Wide residual networks (2016)
23. Zhou, Z., Ong, Y.S., Nair, P.B., et al.: Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **37**(1), 66–76 (2006)