# Principled quality diversity for ensemble classifiers using MAP-Elites

Kyle Nickerson*
Memorial University of Newfoundland

Ting Hu
Queen's University

## ABSTRACT

For many supervised learning tasks, ensemble classifiers – which make predictions by combining multiple simple models – outperform single model classifiers. While genetic programming can be used to evolve populations of simple classifiers, it tends to produce populations of highly similar models. In this work we propose Neuro MAP-Elites (NME) as a method for evolving populations of high performing models which produce diverse predictions, making them suitable for constructing ensembles.

**ACM Reference Format:**
Kyle Nickerson and Ting Hu. 2021. Principled quality diversity for ensemble classifiers using MAP-Elites. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459549

## 1 INTRODUCTION

The importance of diversity in ensemble classifiers is well documented, and while there may be various notions of diversity, for ensemble classifiers it is specifically error diversity which matters most [3].

Genetic programming (GP), an evolutionary algorithm which produces populations of solutions, can be used to create ensembles. However, standard GP algorithms tend to produce populations containing many similar solutions, making them ineffective as ensembles. Recent work in the evolutionary computing community on *quality diversity* [4] has produced methods for increasing population diversity, including the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) algorithm [7], which has been particularly successful at a wide range of tasks.

MAP-Elites represents a population as a structured grid, as opposed to the unstructured set representation used by traditional GP algorithms. In MAP-Elites, candidate solutions only compete with other candidates assigned to the same cell, which allows the population to maintain diversity as evolution proceeds. The mapping from solutions to cells is facilitated by *behavior descriptors*, which map solutions to a low dimensional behavior space. The behavior space is then partitioned into cells, and solutions are assigned to the cells in which their behavior descriptors lie.

Defining effective behavior descriptors is an important aspect of MAP-Elites, as they determine the sorts of diversity which will be produced. Simple behavior descriptors for GP may be based on the complexity of programs (such as the number of operations and features used), or the frequency of different types of operations. However, these are not based on the principle that error diversity is the type of diversity needed in ensembles. The main contribution of this work is a methodology for creating low dimensional descriptors which characterize the behavior of classifiers based on their errors.

*Corresponding author: kln870@mun.ca


*GECCO '21 Companion, July 10–14, 2021, Lille, France*

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3459549

## 2 NEURO-MAP-ELITES

One natural way to represent errors which a classifier makes on a given dataset of $N$ samples is with a length $N$ binary indicator vector $b$, where $\forall i \leq N, b[i] == True$ if and only if the classifier makes an error on the $i^{th}$ sample. The issue with this natural descriptor, is that the dimensionality of the descriptor (indicator vector) will increase linearly with the size of the dataset, whereas MAP-Elites requires low dimensional descriptors, typically 2-3 dimensions.

Our proposed Neuro-MAP-Elites (NME) algorithm operates by finding a 2D encoding of the $N$ dimensional binary error indicator vector which is most informative about the full error vector. The way we measure how informative a 2D encoding is about the original indicator vector is by training a neural network to reconstruct the original vector from the encoding and measuring the quality of the reconstruction. Specifically, NME makes use of the variational autoencoder (VAE) framework [6] to learn optimal encodings for the high-dimensional error vectors.

In our implementation of NME, we used linear genetic programs [1] as the predictive models, however since the behaviour descriptor used in NME is based solely on the predictions it can be used with other GP variants as well. NME consists of 3 phases: mining solutions, VAE training, and MAP-Elites with encoder.

*Mining Solutions.* The goal of the first phase of NME is obtain a sample from the distribution of errors produced by 'good' programs, which is used in phase two to train the VAE. In this work, we consider any solution which is present in a final population and obtains a balanced accuracy score greater than 0.5 on the training data. To create this sample, we run a basic version of MAP-Elites using simple descriptors which count the number of effective instructions and features used, and record the predictions made by each individual in the final populations on all training samples.

*VAE Training.* In the second phase we train a VAE to learn a low-dimensional representation of the errors produced in the first phase. When training the VAE, we use the $\beta$-VAE training objective [5], which contains two terms – one measures the reconstruction quality and the other is a regularization term which encourages the distribution of encodings to be approximately normally distributed – and a constant ($\beta$) which controls the weighting of the two terms. The value of $\beta$ for each dataset was determined using cross validation while training the VAE, from values in the set $\{0.2, 0.3, 0.4\}$.

| Dataset | Classic Machine Learning | | | | | Basic ME | | NME | |
|---|---|---|---|---|---|---|---|---|---|
| | Random Forest | KNeighbors | Logistic Regression | GaussianNB | SVC | Best Single | Vote (PMV) | Best Single | Vote (PMV) |
| Breast | **.950** | .564 | .500 | .949 | .500 | .939 | .579 | .913 | .943 |
| Monk2 | **.987** | .709 | .447 | .456 | .697 | .729 | .502 | .642 | .785 |
| HV_wtihout_noise | .646 | .637 | **1.00** | .523 | .949 | **1.00** | .426 | .986 | **1.00** |
| HV_with_noise | .575 | .520 | **.974** | .526 | .854 | .687 | .495 | .709 | .802 |
| GAMETES_Epistasis | .499 | .515 | .486 | .513 | .486 | .495 | .516 | .510 | **.569** |
| Parity5+5 | .594 | .601 | .463 | .463 | .473 | **1.00** | .494 | .891 | .986 |
| Mean | .709 | .591 | .645 | .572 | .660 | .808 | .502 | .775 | **.848** |

**Table 1: Comparison of balanced accuracy scores of common machine learning classifiers and evolved classifiers. Best scores for each dataset are indicated in bold.**

*MAP Elites with encoder.* In the final phase we run MAP-Elites again, however instead of using the basic descriptors (as in phase 1), we now use the encoder network of the VAE (trained in phase 2) to produce the descriptors. The encoder takes as input a binary prediction vector made by a program on the training data, and outputs a 2-dimensional real valued encoding of the prediction vector. As the encoder was trained as part of a VAE, the distribution of encodings should be approximately a unit normal distribution. We take advantage of this fact when partitioning the latent space into bins to use with MAP-Elites and set the bin boundaries so that each bin has equal probability mass under a normal distribution.

## 3 EXPERIMENTS AND RESULTS

To demonstrate the efficacy of our method, we compare the performance achieved by classifiers generated with NME against the performance of standard machine learning classifiers, as well as genetic programming classifiers evolved using a classic version of MAP-Elites. All experiments using standard classifiers used the classifier implementations from scikit-learn [9], as well as scikit-learn utilities for tuning their parameters[1].

For both the basic MAP-Elites and NME we tested 3 ways of creating a final classifier from the final population. The *best single* classifier selects a single best program from the final population, and this program is used for as the final classifier. The *full majority vote* (FMV) classifier uses the entire population, and outputs the prediction of the majority of the programs. The *partial majority vote* (PMV) employs a scheme similar to the FMV, but only includes programs with fitness about a threshold $t$, where $t$ is chosen to maximize the accuracy of the majority vote classifier on the training data. In our experiments the PMV classifier always outperformed the FMV version, so results from FMV have been omitted.

*Dataset Selection.* As our algorithm produces and ensemble classifier, we are particularly interested in how it preforms relative to other ensemble classifiers. To this end we selected the datasets based on the performance of a common ensemble classifier, random forest [2]. We use a subset of datasets from the Penn Machine Learning Benchmarks (PMLB) repository [8]. Specifically, we selected two datasets where random forest performs much better than other standard classifiers, two datasets where random forest performs much worse and finally two datasets where all the tested standard algorithms do poorly.

[1]see github.com/bigtuna08/nme/ for the code to tune parameters of all models

*Comparison Results.* Measured across all datasets, our method compares favorably against both the traditional machine learning classifiers and LGP classifiers evolved using MAP-Elites with basic LGP descriptors, however no method is a clear winner across all of the datasets (see Table 1). On the two datasets which were selected for being easy for random forest (Breast and Monk2), random forest was the most accurate. On one of the datasets which was significantly harder for random forest than other standard classifiers (HV_without_noise) multiple methods achieved perfect accuracy, including NME partial vote; on the other (HV_with_noise), multiple standard methods outperform all evolved classifiers. Finally, on the datasets which were difficult for standard methods (Parity5+5 and GAMETES_Epistasis), evolved classifiers outperformed the standard ones, although in only one of these (GAMETES_Epistasis) was the NME classifier the best.

Another finding evident from Table 1 is that of the methods test that evolve ensembles, those created with NME significantly outperform ensembles created with the basic variants of MAP-Elites, despite the fact that basic MAP-Elites can produce high quality single solutions, sometimes better than the singles solutions produced by NME. This result provides support for our hypothesis that NME generates populations with more meaningful diversity, and that this diversity is beneficial for creating ensembles.

## REFERENCES


[1] Markus F. Brameier and Wolfgang Banzhaf. 2007. *Linear Genetic Programming* (1st ed.). Springer Publishing Company, Incorporated.

[2] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45, 1 (Oct. 2001), 5–32.

[3] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. 2005. Diversity creation methods: a survey and categorisation. *Information Fusion* 6, 1 (2005), 5 – 20. Diversity in Multiple Classifier Systems.

[4] A. Cully and Y. Demiris. 2018. Quality and Diversity Optimization: A Unifying Modular Framework. *IEEE Transactions on Evolutionary Computation* 22, 2 (April 2018), 245–259.

[5] I. Higgins, Loïc Matthey, A. Pal, C. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*.

[6] Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. (2014). arXiv:stat.ML/1312.6114

[7] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv* arXiv/1504.04909 (2015).

[8] Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. 2017. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining* 10, 1 (11 Dec 2017), 36.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.