A Significantly Faster Elastic Ensemble for Time Series Classification

George Oastler and Jason Lines

University of East Anglia, Norwich, United Kingdom
{g.oastler, j.lines}@uea.ac.uk

Abstract. We investigate two simple and intuitive strategies for reducing the training effort required to build the Elastic Ensemble for time series classification. First, we reduce the number of parameter options available to each constituent classifier in training and observe the effect that this has on test performance. Second, we limit the number of training series used when optimising parameters through train set estimates to reduce build times. Through initial experimentation over 10 folds of 48 time series classification problems we find that both approaches separately lead to significantly faster build times without significant loss in accuracy, and crucially, combining both of these transparent and simple speed-ups leads to even greater speedup without loss in accuracy.

Keywords: time series · classification · ensembles

1 Introduction

The current state of the art in time series classification (TSC) research is the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) [8]. One of the five constituent ensembles within HIVE-COTE is the Elastic Ensemble (EE) [7]; EE is an ensemble of eleven nearest-neighbour (NN) classifiers that are each coupled with *elastic* distance measures. A recent experimental evaluation [1] demonstrated that EE is a leading TSC algorithm in its own right, but it is also key within HIVE-COTE to allow the meta ensemble to uncover discriminatory features of TSC problems that exist in the time-domain. However, a weakness of the algorithm is the time complexity required to build the classifier. Ten of the constituent classifiers use distance measures with $O(m^2)$ run-time complexities, and eight of these require parameters to be set. This is done through considering 100 parameter options for each, set through leave-one-out cross-validation (LOOCV) on the training data. EE therefore does not scale well with very large datasets and is currently a bottleneck in the build time of HIVE-COTE.

This work forms part of an ongoing effort to increase the utility of HIVE-COTE for very large problems. We are interested in investigating simple strategies for accelerating the training of EE with a view to contracting the classifier to run within a predetermined time limit. Distance measures have been long studied within TSC research and there exist various applicable measure-specific

speed-ups such as distance measure lower-bounds. However, in addition to this, two simple and practical training adjustments are:

1. using less cases when estimating the relative accuracy of different parameter options in training, and
2. using less parameter options when training constituent classifiers.

These simple strategies are transparent and clear, but it is unclear without further investigation what the implications would be in terms of test accuracy.

Our hypotheses are two-fold: first, we believe that the relative performance of different parameter options can be discerned through a much smaller set of training data than what is required for test classification phase. Therefore we believe we can optimise parameter options much faster through comparing relative performance on fewer neighbours without sacrificing test accuracy. Second, we believe that there is a high degree of overlap between the performance of similar parameter options for constituent classifiers, and therefore using a sufficiently large subset of parameter options would not lead to a significant drop in test accuracy.

We initially design two experiments to test our hypotheses using variants of EE built over 10 resamples of 48 datasets compared against a full EE using all training data and all default parameter options, as outlined in [7]. First we use the full EE and compare against versions that use 10% or 50% of the available training data (randomly selected) to optimise distance measure parameters. We arbitrarily selected 10% and 50% in advance to avoid biasing values towards these datasets and note that these values may not be optimal. Second, we use full training data to compare parameter options, but this time limit the number of parameter options, again using the values of 10% and 50% to randomly select from the pool of possible model selections. Finally, we design a subsequent experiment to combine these strategies and test the viability of sampling both parameter options and training data used to optimise them.

The results demonstrate that significant reductions in training time are possible without any significant loss in test accuracy through either limiting the number of parameters or limiting the amount of training data used to optimise parameters. Crucially, the follow-up experiment also demonstrates that it is possible to limit both parameters and neighbours without significantly affecting the test accuracy of EE. We conclude that it is possible to speed up EE, on average, by over two orders of magnitude in training without any sacrifice in test accuracy through utilising the simple training strategies that we outline in this work.

2 Background and Related Work

A time series is a sequence of real-valued, ordered data that is typically ordered by time. For the objective of TSC, we require a time series dataset $\mathbf{T} = \{T_1, T_2, ..., T_n\}$ where each time series consists of m ordered observations, $T_i =< t_1, t_2, ..., t_m >$ (where m is the length of the time series), and a class label c_i. The objective of TSC is to use a set of training data to create a function that

maps from the space of possible time series to the space of possible class labels to classify unseen test cases.

TSC is an active area of research where many diverse algorithms have been proposed. These include, but are not limited to, histogram-based approaches that discriminate cases based on the frequency of reoccurring patterns [10,11]; shapelet algorithms that differentiate class membership through the presence of discriminatory, phase-independent subsequences [13,4]; and forest ensembles built on data transformed into different representations [8,3]. Arguably, most TSC research effort over the last decade has been focused on developing *elastic* distance measures to couple with simple 1-nearest-neighbour (1-NN) classifiers. Such elastic distance measures are able to mitigate misalignments and phase-shift within data, and are key for classifying TSC problems where discriminatory features are found in the time-domain. The most common approach, and long considered gold-standard of TSC, is to use dynamic time warping (DTW) with a warping window set through cross-validation and a 1-NN classifier. Related variants of DTW have also been proposed, such as applying a soft boundaries to warping windows through weighting penalties [5], and warping directly on first-order derivatives [6]. Alternatives exist that are derived from the edit-distance [2], and further hybrid measures have characteristics of both DTW and edit-distance [9,12].

2.1 The Elastic Ensemble

The performance of alternative elastic distance measures with 1-NN classifiers were compared in [7] to determine whether one approach outperformed all others. The measures included were: dynamic time warping (DTW), derivative DTW (DDTW), weighted variants of both DTW and DDTW (WDTW, WDDTW), edit distance with real penalty (ERP), longest common subsequence (LCSS), time warp edit (TWE) and move-split-merge (MSM). All measures were coupled with 1-NN classifiers and had 100 model selections in training to optimise distance-measure parameters. Euclidean distance, full-window DTW and full-window DDTW were used as baselines and all 11 subsequent 1-NN classifiers were compared over 85 TSC datasets. It was found that no single measure significantly outperformed all others in terms of test accuracy. However, it was found that the classifiers did make significantly *different* predictions. This diversity in choice was the inspiration behind the Elastic Ensemble (EE), an ensemble of the 11 1-NN classifiers described above. In training, each constituent is evaluated using a LOOCV in training to obtain both the optimal parameter option for that classifier and an estimate of test set accuracy. In testing, each constituent is given a vote weighted by its training accuracy estimate and the ensemble predicts the class with the greatest weighted vote.

3 Proposed Enhancements

3.1 Limit the neighbourhood size

For every constituent within EE, each distance-measure parameter is compared using LOOCV experiments with full training data to select the best candidate parameter setting for subsequent test classification. Our first hypothesis is that we believe it is unnecessary to use all training data to uncover the relative ordering of parameter options. We believe a much smaller subset can be used for model selection as the relative ranking of model selections will be very similar whether they are evaluated on 100% of the training data or a suitably large subset of it. The key is to determine how much data is required for this; for our first training strategy we arbitrarily pick 10% and 50% for the purpose of this investigation, but note that it may be possible to dynamically determine this through incrementally adding training data until no changes in the ranks of parameters occur. Once the best parameters are selected, the classifier can be trained with the full training data to provide a training weight to EE and make test predictions using the full training neighbourhood. We outline the generic form of this training strategy in algorithm 1 line 1 to 5.

3.2 Limit the parameter space size

Each distance measure within EE has 100 possible distance measure parameters. Searching for the optimal parameter setting is an iterative process and we believe a large proportion of the parameters are redundant due to adjacent options yielding same or very similar test predictions. Our second proposed training strategy is to randomly sample parameter options rather than using an exhaustive search of 100 options. In our experiments we arbitrarily use 10% and 50% and use the sampling procedure outlined in algorithm 1 line 7 to 25.

3.3 Combined Strategies

A third strategy is to combine both strategies described in Section 3.1 and Section 3.2 to limit the range of neighbours and the number of parameters used to compare options in training. The feasibility of this approach is clearly dependent on the first and second training strategies. We design a follow-up experiment to investigate the combination of these approaches and observe the results for all combinations of 10% and 50% for parameter and neighbours against 100% to be consistent with our previous strategies.

4 Experimental Design

We design experiments to test the feasibility of the three EE training strategies outlined in Section 3. These experiments are part of an ongoing effort and are currently reported over 10 resamples of a subset of 48 datasets due to computational and time constraints, demonstrating the need for a faster EE. These

Algorithm 1 buildElasticEnsemble(**T**, n, m, p, q, **M**) **return** N

Require: $n \times m$ List **T**, data-set of n time series of length m
Require: p numbers of parameters per distance measure
Require: q number of neighbours
Require: **M** array of distance measure names

$R \leftarrow \emptyset$ ▷ *initialise reduced train-set to list of q time series of length m*
for $i \leftarrow 1$ to q **do** ▷ *choose a subset of the neighbours*
  $r \leftarrow randint(q)$ ▷ *uniform random integer*
  $t \leftarrow remove(T, r)$ ▷ *remove time series from train-set*
  $R \leftarrow R \cup t$ ▷ *add time series to reduced train-set*
$T \leftarrow R$ ▷ *tune distance measure parameters on reduced train-set*
for $i \leftarrow 1$ to $len(M)$ **do** ▷ *for each distance measure*
  $dm \leftarrow M[i]$
  $paramSpace \leftarrow getParamSpace(dm, T)$ ▷ *get the parameter space for the distance measure using the train-set*
  $bestParams \leftarrow \emptyset$
  $bestParam \leftarrow paramSpace[0]$
  $bestAcc \leftarrow evalParam(dm, bestParam, \mathbf{T})$
  $bestParams \leftarrow bestParams \cup bestParam$
  for $j \leftarrow 2$ to p **do**
    $param \leftarrow paramSpace[j]$
    $acc \leftarrow evalParam(dm, param, \mathbf{T})$
    if $acc >= bestAcc$ **then** ▷ *keep multiple parameter if they have the same accuracy*
      if $acc > bestAcc$ **then**
        $bestParams \leftarrow \emptyset$
        $bestParam \leftarrow param$
        $bestAcc \leftarrow acc$
      $bestParams \leftarrow bestParams \cup bestParam$
  $r \leftarrow randint(len(bestParams))$ ▷ *randomly choose from the best parameters*
  $bestParam \leftarrow bestParams[r]$
  $setParam(dm, bestParam)$

results are indicative of the performance of the training strategies however, and further results will be added in due course when available to confirm the findings over the complete repository. A full list of datasets that we have completed results for is given in appendix 1 and all source code to generate the results can be downloaded from the provided link [1].

5 Results

The following results are organised into three separate experiments. First, we report finding of optimising parameters with less data; second, we report findings of reducing the number of parameters in training; and finally we report

[1] UEA-TSC Git Repository

results for combining both of the previous strategies. For each experiment we report accuracies to validate the feasibility of the strategy and timing results to underline the best fit of each in practice.

The critical difference (CD) diagrams used throughout these experiments visually demonstrate the results of comparing all classifiers using pairwise Wilcoxon signed rank tests, where *cliques* are formed using the Holm correction to represent classifiers where there is no significant difference between them.

5.1 Reducing cases for parameter optimisation

The results of the first experiment are summarised in the CD diagram in Figure 1 and full results are available in Table 1.

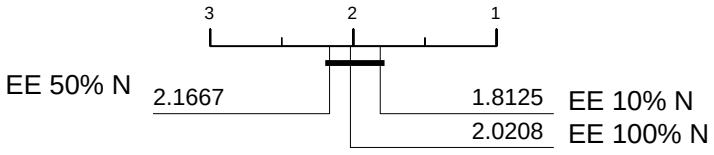


Fig. 1: A pairwise critical difference diagram to compare the test accuracy of EE using various amounts of training data to optimise parameter settings. For clarity, 10% N uses 10% of the available training data to optimise measure parameters in training.

The results demonstrate that there is no significant pairwise difference in terms of accuracy between using 10% or 50% of the training data when compared to using 100% of the training data to optimise parameters. This confirms our hypothesis that the relative rankings of the parameter options for constituents of EE holds, even when using much fewer data to establish the rankings. This demonstrates that a significant train-time saving is possible through comparing parameters on less neighbours, as shown in Table 2. We further demonstrate the equivalence of using 10% versus 100% of neighbours when optimising parameters with the accuracies over the 48 datasets in the scatter plot of Figure 2.

5.2 Reducing parameter options in training

The results of the second experiment are summarised in the CD diagram in Figure 3 and full results are available in Table 1.

The results of the second experiment are less clear-cut but the message is the same; using 10% of the possible parameter options in training for EE constituents is not significantly different to the original EE with 100% of parameter options in terms of accuracy. This confirms that training of EE can be accelerated simply by reducing parameter options without significantly sacrificing test accuracy. Interestingly there is a significant difference in favour of 50% over 10%. This

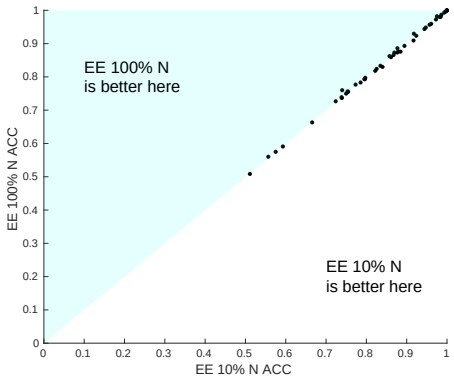


Fig. 2: A scatter plot to compare test accuracies between full EE and a variant where only 10% of the training data is used in model selection. Results in the top-half are where datasets perform best with the full EE, while results on the diagonal indicate identical performance on that given dataset.

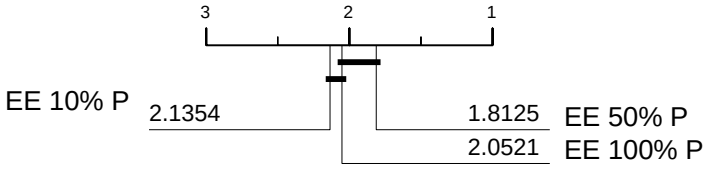
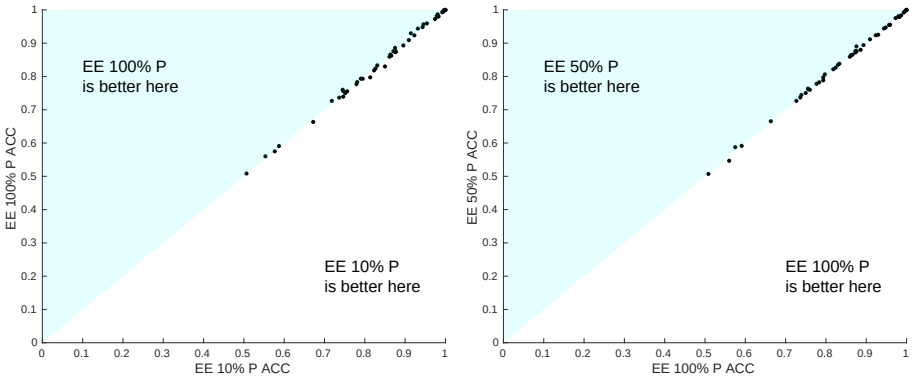


Fig. 3: A pairwise CD diagram to compare the test classification performance of EE using various amounts of available parameter options in training. For clarity, 10% P is retricted to only consider 10% of the possible parameter values for each constituent classifier when compared to the full EE.

suggests that randomly sampling only 10% of the parameter options may be too few for at least a subset of the constituent classifiers within EE and further investigation is required. We provide side-by-side scatter plots of 10% and 50% parameters versus 100% in figure 4 to further demonstrate the results where there were no significant difference in terms of test accuracy.

5.3 Combined strategy: reducing neighbours and parameter options

The results presented in Sections 5.1 and 5.2 provide an opportunity to further accelerate the training of EE. To reiterate, in terms of test set accuracy over 10 resamples of 48 TSC problems, there is no significant pairwise difference between the full EE algorithm versus reducing training data to 10% for optimisation of distance-measure parameters, and no significant difference between considering 100% or 10% of possible parameter options in training. The interesting question therefore is *can we combine both strategies without a significant sacrifice in test accuracy*? To remain consistent with the previous experiments



(a) 10% vs. 100% (b) 50% vs. 100%

Fig. 4: Scatter plots to compare test accuracies between full EE and two variants that are given different proportions of possible parameter options in training: (a) compares full EE to EE with only 10% of parameter values and (b) compares full EE to having 50% of parameter options available in training.

we again arbitrarily use the values of 10% and 50%, using the four combinations of each (10%/10%, 10%/50%, 50%/10%, and 50%/50%) and comparing to the full EE with 100% of available neighbours and parameter options. These results are summarised in the critical difference diagram in Figure 5.

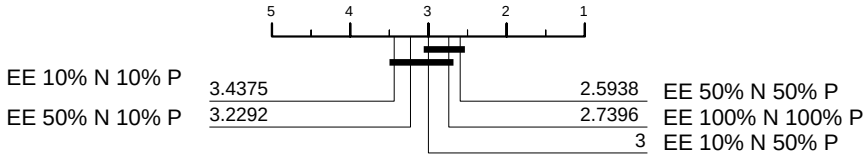


Fig. 5: A pairwise critical difference diagram to compare the test accuracy of EE built using various amounts of training data to assess model selections (N) and limited amounts of candidate parameter values (P). For example, 50% N and 10% P uses 50% of the available training data to assess 10% of the possible parameter options that the full EE uses.

The results in Figure 5 confirm that there is no pairwise significant difference in the test accuracies of full EE and EE trained using only 10% of parameter options and 10% of training data to optimise the parameter options (EE-10%). This training heuristic of course does not formally lower the order of the run-time complexity of EE, but in practical terms, the same test accuracy is achieved

by EE-10% while only requiring 3% of the time to build the classifier on average (full timing results are available in Table 2).

It should be noted however that, while not significantly different to full EE, the test predictions of EE-10% are significantly less accurate than the best performing variant of EE overall which uses 50% of training data and 50% of parameters. The results indicate again that conservative sampling of the parameter space may be wise, but the timing results in Table 2 indicate that the best speed-ups in training are obtained from using less neighbours to evaluate parameter options. Therefore our advice to practitioners is that using 50% of parameter options and 10% of training data to evaluate them seems to be a good compromise in terms of test accuracy and classifier build time. The EE with this configuration was ranked 3rd overall in figure 5 and was not significantly outperformed by any other training strategy. Further, the timing results in figure 2 show that, on average, this EE setup can also be trained in approximately only 6.6% of the time of the full EE. This is over an order of magnitude faster than the original algorithm. We demonstrate the parity in test performances between this recommended strategy and the full EE in figure 6.

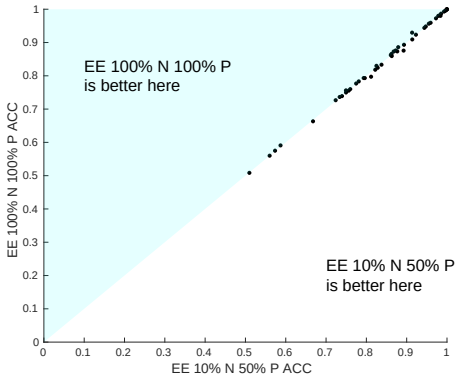


Fig. 6: Pairwise scatter diagram comparing test accuracy of full EE against the recommended EE configuration.

6 Conclusions and future work

In this work we have proposed two strategies for reducing the time required to build EE for TSC problems. First, we described a simple approach to use less training cases when optimising the distance-measure parameters of the constituent classifiers. Second, we described another simple strategy to reduce the parameter search space through random selection of method parameters on the basis that similar parameters values yield very similar test predictions. We hypothesised that both could lead to faster build times of EE without significantly

affecting test accuracy. We validated these claims by performing experiments that demonstrated using either 10% of neighbours or 10% of parameters both resulted in an EE classifier that was not significantly outperformed by the full EE over 10 resamples of 48 TSC problems.

Inspired by these findings, we combined these two strategies to build EE classifiers with less parameter options and less training data used to optimise them. We found that using 10% values for both in combination only took approximately 3% of the time that the full EE required to build while producing test results that were not significantly different.

References

1. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **31**(3), 606–660 (2017)
2. Chen, L., Ng, R.: On the marriage of Lp-norms and edit distance. In: Proc. 30th International Conference on Very Large Databases (VLDB) (2004)
3. Deng, H., Runger, G., Tuv, E., Vladimir, M.: A time series forest for classification and feature extraction. Information Sciences **239**, 142–153 (2013)
4. Hills, J., Lines, J., Baranauskas, E., Mapp, J., Bagnall, A.: Classification of time series by shapelet transformation. Data Mining and Knowledge Discovery **28**(4), 851–881 (2014)
5. Jeong, Y., Jeong, M., Omitaomu, O.: Weighted dynamic time warping for time series classification. Pattern Recognition **44**, 2231–2240 (2011)
6. Keogh, E., Pazzani, M.: Derivative dynamic time warping. In: Proc. 1st SIAM International Conference on Data Mining (SDM) (2001)
7. Lines, J., Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery **29**, 565–592 (2015)
8. Lines, J., Taylor, S., Bagnall, A.: Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. ACM Trans. Knowledge Discovery from Data **12**(5) (2018)
9. Marteau, P.: Time warp edit distance with stiffness adjustment for time series matching. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(2), 306–318 (2009)
10. Schäfer, P.: The BOSS is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery **29**(6), 1505–1530 (2015)
11. Schäfer, P., Leser, U.: Fast and accurate time series classification with weasel. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 637–646. ACM (2017)
12. Stefan, A., Athitsos, V., Das, G.: The Move-Split-Merge metric for time series. IEEE Transactions on Knowledge and Data Engineering **25**(6), 1425–1438 (2013)
13. Ye, L., Keogh, E.: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Mining and Knowledge Discovery **22**(1-2), 149–182 (2011)

7 Appendix

Table 1: Table of accuracies over the 48 datasets comparing EE with different configurations of neighbours and parameters. N corresponds to percentage of neighbours, P corresponds to the percentage of parameters.

Dataset	100% N 100% P (minutes)	50% N 50% P (%)	10% N 50% P (%)	50% N 10% P (%)	10% N 10% P (%)
ArrowHead	0.8657	0.8646	0.8623	0.8646	**0.8703**
Beef	**0.5600**	0.5533	**0.5600**	0.5567	**0.5600**
BeetleFly	0.7600	**0.7650**	0.7600	0.7500	0.7600
BirdChicken	0.8300	0.8400	0.8250	0.8400	**0.8450**
BME	**0.9727**	0.9713	**0.9727**	0.9700	0.9680
Car	**0.7933**	0.7917	**0.7933**	0.7800	0.7817
CBF	0.9801	**0.9840**	0.9839	0.9834	0.9799
Chinatown	0.9235	0.9214	0.9229	**0.9241**	0.9223
Coffee	**0.9821**	0.9786	**0.9821**	0.9786	0.9750
DistPhlnxOAG	0.7554	0.7547	**0.7561**	0.7547	0.7540
DistPhlnxOC	0.7768	0.7764	0.7746	0.7786	**0.7815**
DistPhlnxTW	0.6633	0.6669	0.6676	0.6698	**0.6727**
ECG200	**0.8860**	0.8830	0.8790	0.8740	0.8700
ECG5Days	0.8333	**0.8388**	0.8375	0.8343	0.8318
FaceFour	0.8761	0.8830	0.8727	**0.8852**	0.8750
GunPoint	**0.9593**	0.9560	**0.9593**	0.9553	0.9540
GunPointAS	0.9953	0.9946	0.9959	**0.9962**	0.9956
GunPointMvF	0.9991	0.9991	0.9991	**0.9994**	**0.9994**
GunPointOvY	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Ham	**0.7267**	0.7257	0.7238	0.7190	0.7190
Herring	0.5750	**0.5844**	0.5734	**0.5844**	0.5813
ItalyPower	**0.9483**	0.9480	0.9474	0.9425	0.9430
Lightning2	0.8623	0.8590	0.8607	0.8607	**0.8639**
Lightning7	0.7562	**0.7575**	0.7493	0.7534	0.7521
Meat	**0.9800**	**0.9800**	0.9783	**0.9800**	0.9783
MedImgs	0.7499	0.7503	0.7495	0.7497	**0.7505**
MelPedest	0.9440	**0.9445**	0.9438	0.9318	0.9316
MidPhlnxOAG	**0.5909**	**0.5909**	0.5870	0.5883	0.5883
MidPhlnxOC	**0.7832**	0.7818	0.7811	0.7794	0.7787
MidPhlnxTW	0.5084	0.5078	**0.5097**	0.5052	0.5000
MoteStrain	**0.8730**	0.8678	0.8677	0.8717	0.8712
OliveOil	0.8733	**0.8767**	**0.8767**	**0.8767**	**0.8767**
Plane	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
PowerCons	**0.9567**	0.9556	0.9550	0.9472	0.9456
ProxPhlnxOAG	0.7932	**0.7995**	0.7961	0.7966	0.7951
ProxPhlnxOC	0.8182	0.8206	**0.8220**	0.8206	0.8216
ProxPhlnxTW	0.7390	0.7459	0.7395	**0.7473**	0.7463
ShapeletSim	0.7972	0.8078	0.8117	**0.8150**	0.8078
SmthSubspace	**0.9867**	0.9840	0.9853	0.9813	0.9800
SonyAIBOS1	0.8241	**0.8295**	0.8281	0.8270	0.8275
SonyAIBOS2	0.8932	0.8924	0.8933	**0.8958**	0.8951
SynthCtrl	**0.9930**	0.9927	0.9927	0.9923	0.9920
ToeSeg1	0.7364	**0.7395**	0.7338	0.7342	0.7351
ToeSeg2	0.9092	0.9123	**0.9138**	0.9100	0.9123
Trace	**0.9990**	**0.9990**	**0.9990**	0.9950	0.9940
TwoLeadECG	**0.9297**	0.9237	0.9134	0.9117	0.9207
UMD	0.8757	**0.8965**	0.8924	0.8688	0.8694
Wine	0.8593	**0.8630**	**0.8630**	0.8574	0.8593
Mean	0.8478	**0.8491**	0.8477	0.8466	0.8465

Table 2: Table comparing the original runtime of EE, in minutes, against the relative improvement for different configurations of percentage of neighbours and percentage of parameters. N corresponds to percentage of neighbours, P corresponds to the percentage of parameters.

Dataset	100% N 100% P (minutes)	50% N 50% P (%)	10% N 50% P (%)	50% N 10% P (%)	10% N 10% P (%)
ArrowHead	19.2656	0.4366	0.0219	0.2464	0.0091
Beef	26.5728	0.5393	0.0387	0.4907	0.0173
BeetleFly	32.0891	0.4769	0.0217	0.4307	0.0076
BirdChicken	30.1739	0.4563	0.0232	0.4727	0.0139
BME	29.5845	0.4106	0.0062	0.3548	0.0034
Car	44.5560	0.4514	0.1247	0.3651	0.0473
CBF	21.7839	0.4311	0.0084	0.3956	0.0071
Chinatown	21.2060	0.7179	0.0224	0.2195	0.0008
Coffee	22.9984	0.5894	0.0180	0.4579	0.0084
DistPhlnxOAG	44.7636	0.4658	0.1322	0.3872	0.0559
DistPhlnxOC	77.7288	0.4951	0.1738	0.3967	0.0705
DistPhlnxTW	47.3070	0.4987	0.1310	0.3950	0.0533
ECG200	25.1638	0.5299	0.0293	0.3653	0.0106
ECG5Days	15.7720	0.5238	0.0068	0.6351	0.0078
FaceFour	26.6080	0.4357	0.0159	0.5105	0.0063
GunPoint	25.1393	0.4989	0.0143	0.4532	0.0098
GunPointAS	32.9800	0.4642	0.0586	0.4454	0.0239
GunPointMvF	34.3784	0.4403	0.0566	0.4211	0.0239
GunPointOvY	32.2634	0.4456	0.0595	0.4324	0.0237
Ham	65.7508	0.4444	0.1565	0.4092	0.0587
Herring	41.9376	0.4668	0.1158	0.3828	0.0468
ItalyPower	28.2579	0.4574	0.0071	0.4717	0.0043
Lightning2	57.6153	0.4537	0.1373	0.3773	0.0530
Lightning7	36.1584	0.5642	0.0797	0.4339	0.0304
Meat	35.5883	0.4625	0.0951	0.3754	0.0359
MedImgs	61.6399	0.5100	0.1234	0.4438	0.0464
MelPedest	93.0124	0.4683	0.0763	0.4539	0.0408
MidPhlnxOAG	47.6292	0.4533	0.1279	0.3716	0.0535
MidPhlnxOC	80.2390	0.4145	0.1672	0.3760	0.0657
MidPhlnxTW	49.2935	0.4285	0.1191	0.3466	0.0505
MoteStrain	15.1864	0.5826	0.0068	0.3962	0.0013
OliveOil	33.0126	0.4634	0.0445	0.4041	0.0159
Plane	31.6079	0.4632	0.0399	0.4515	0.0192
PowerCons	40.0665	0.4636	0.1037	0.4022	0.0390
ProxPhlnxOAG	46.1836	0.4381	0.1252	0.4031	0.0491
ProxPhlnxOC	76.2361	0.5068	0.1761	0.3529	0.0698
ProxPhlnxTW	45.3355	0.4951	0.1315	0.3933	0.0527
ShapeletSim	32.5059	0.4307	0.0314	0.3858	0.0126
SmthSubspace	30.2103	0.4127	0.0112	0.3185	0.0088
SonyAIBOS1	16.4059	0.5773	0.0069	0.5387	0.0017
SonyAIBOS2	16.8875	0.4785	0.0026	0.3917	0.0015
SynthCtrl	35.4965	0.5216	0.0730	0.5080	0.0305
ToeSeg1	20.9472	0.4662	0.0447	0.3359	0.0143
ToeSeg2	25.5469	0.4792	0.0299	0.5161	0.0124
Trace	38.0855	0.4729	0.1101	0.4057	0.0401
TwoLeadECG	13.5956	0.4254	0.0033	0.2039	0.0017
UMD	26.0158	0.4387	0.0093	0.4880	0.0109
Wine	30.9764	0.4848	0.0268	0.4072	0.0162
Mean	37.1200	0.4798	0.0655	0.4088	0.0268