# A Factorial Study of Neural Network Learning from Differences for Regression

Mathieu D'aquin, Emmanuel Nauer, Jean Lieber

# A factorial study of neural network learning from differences for regression

Mathieu d'Aquin, Emmanuel Nauer, and Jean Lieber

LORIA, Université de Lorraine, CNRS, INRIA,
Vandœuvre-lès-Nancy, France
firstname.lastname@loria.fr

**Abstract.** For regression tasks, using neural networks in a supervised way typically requires to repeatedly (over several iterations called epochs) present a set of items described by a number of features and the expected value to the network, so that it can learn to predict those values from those features. Inspired by case-based reasoning, several previous studies have made the hypothesis that there could be some advantages in training such neural networks on differences between sets of features, to predict differences between values. To test such a hypothesis, we applied a systematic factorial study on seven datasets and variants of datasets. The goal is to understand the impact on the performance of a neural network trained on differences, as compared to one trained in the usual way, of parameters such as the size of the training set, the number of epochs of training or the number of similar cases retrieved. We find that learning from differences achieves similar or better results than the ones of a neural network trained in the usual way. Our most significant finding however is that, in all cases, difference-based networks start obtaining good results from a low number of epochs, compared to the one required by a neural network trained in the usual manner. In other words, they achieve similar results while requiring less training.

**Keywords:** case-based reasoning, neural network, case difference heuristic, learning from differences, factorial study

## 1 Introduction

As described in more details in the related work section of this paper (Section 2), the idea of learning from differences in regression is not new. Inspired by case-based reasoning (CBR), and viewed for example in [4] as a way to acquire adaptation knowledge, the idea is to estimate an unknown value (the solution) associated to a target set of features (the problem) by predicting its difference with the value associated with a known, similar source case from the training set (the case base). From a machine learning point of view, this corresponds to training a model (here we focus on neural network models) with differences of features, to predict differences in target values.

As an example, based on a real dataset used in the experiments later in this paper, while a neural network trained in the usual way would try to predict the

sale price of a used car directly based on information such as its age, mileage, or engine size, the approach of learning from differences consists in building a similar neural network, but that is able to predict the differences in prices between similar cars, given the differences in their age, mileage or engine size. Such a network can therefore be used in a process based on the CBR methodology, predicting the price difference between the car under consideration and a similar car retrieved from the case base/training set.

While doing so does not formally increase the amount of information provided to the neural network for learning, it can be expected to have a number of advantages. At a meta-level, the results might be more interpretable, since the prediction of the price of a used car, in our example, might be easier to explain in reference to a similar car for which the price is known. What we consider here however is of a different nature: We want to test the hypothesis that CBR-inspired learning from differences might have advantages with respect to the training performance of neural network models, and under what circumstances (size of the data, training time, number of similar cases used) those advantages might materialise.

To achieve this, we set up a factorial study on seven datasets and variants of datasets. By factorial study we mean that we systematically train and measure the performance of neural networks in the usual way (using the original features of the data) and using case differences, while varying a number of factors: number of epochs, size of the testing and training sets, etc. This allows us to check in which way those factors influence the performance of the networks trained in those two ways. The contributions of this paper therefore include:

- A publicly available python library facilitating the process of training neural networks (and possibly other machine learning models) from case differences under various parameters,
- a large set of performance results (also publicly available) from the thousands of models trained under different sets of parameters,
- the main findings that a key effect of learning from differences requires significantly less epochs (from half to two orders of magnitude less) to reach the same/similar performance results as a neural network trained in the usual way, and
- a report on which parameter values in the configuration of the training and prediction processes achieve the best performance on the selected seven datasets and variants.

We start by describing related works on learning from differences in the next section. We then describe the process we applied and the parameters that have been used as varying factors in our factorial study in Section 3. We also describe the methodology for the factorial study in Section 4 and the results of applying this methodology in Section 5. Finally, we conclude on the main findings of this study in Section 6.

## 2 Related Work

Many works have addressed the use of neural network in a CBR process, especially for acquiring adaptation knowledge. These works are based on the exploitation of the case base, following the idea that adaptation knowledge can be acquired from differences between pairs of cases using the case difference heuristic (CDH) [2]. The CDH has been applied first outside of the context of neural networks to produce adaptation rules with various approaches [2,3,6].

More recently, neural networks have been used with CDH to learn the differences between solutions from the differences between problems [5,4,8]. In these works, particular points have been studied. [5], which is a preliminary work, shows the feasibility of using a neural network to correct the solution of the most similar retrieved case. The impact of the size of the training set is also studied, showing, unsurprisingly, that using more cases (80% of the case base) works better than using less cases (20%).

[4] compares 5 systems: nearest neighbour (1-NN) retrieval with copy adaptation, average of the solutions of a 3-NN process (i.e. $k$ Nearest Neighbours, kNN, with $k = 3$), a classical neural network which takes problem features as input to predict the solution, a CBR system inspired from [1] using adaptation rules generated with CDH in addition to the problem features as context to adapt the most similar case, a CBR system which adapts the most similar case using a neural network which has learned how to adapt with CDH (i.e. from differences). This study concludes that the last system outperforms the two first ones (without adaptation) and outperforms the system which adapts by rules, being only outperformed by the neural network which solves the regression problem directly. This is the closest work to our own, since it provides some elements of comparison between neural networks learning from differences (last system), and learning in the usual way (third system). However, this study addresses a very particular issue: solving *novel* problems, by removing a certain number of cases that are the most similar to the target problem (sometimes up to a third of the case base), and does not therefore report on factors such as the ones studied in the present paper.

To the best of our knowledge, no work has really examined in details the impact of parameters usually involved in CBR experiments (case base size for training, number of cases retrieved for prediction, etc.) in the framework of using neural network with CDH for case adaptation. Additionally, as the neural network is central to the process, some parameters of its learning process (e.g. number of epochs) could also play an important role and deserve to be considered.

## 3 Learning from differences for regression

We consider here a dataset as a set $D$ of pairs $(X_i, y_i)$ where $X_i$ is a vector of $n$ numerical values corresponding to the features of a given example in the dataset, and $y_i$ is a value corresponding to the result we expect to be able to predict.

In the example of the cars, a given set of features $X_i$ includes for example the brand, model,[1] engine size and mileage, and the corresponding value $y_i$ is the price at which that car was sold.

Considering this, the typical process to train and validate a neural network is the following:

1. Split the dataset $D$ into two subsets $D^{train} = \left\{(X_i^{train}, y_i^{train})\right\}_i$ and $D^{test} = \left\{(X_j^{test}, y_j^{test})\right\}_j$ through random sampling.
2. Fit the neural network to $D^{train}$ over a number of iterations *epochs*.
3. Use the trained neural network to predict the value corresponding to each $X_j^{test}$; the predicted value is denoted by $y_j^{pred}$.
4. Compare $y_j^{pred}$ to $y_j^{test}$ to assess the accuracy of the predictions (in our experiments, we use the $\mathbf{R^2}$ measure).

We do not detail those steps since they are relatively standard and are implemented using commonly used libraries (in our experiments, keras[2] and scikitlearn[3]).

Learning from differences includes similar steps, with additional elements to prepossess the data in order to transform it into sets of case differences, and to compute predictions of actual values from predicted case differences. In more details, the process involves the following:

1. Split the dataset $D$ into two subsets $D^{train} = \left\{(X_i^{train}, y_i^{train})\right\}_i$ and $D^{test} = \left\{(X_j^{test}, y_j^{test})\right\}_j$ through random sampling.
2. For each case $C_i = (X_i^{train}, y_i^{train}) \in D^{train}$, retrieve $ntr$ similar cases from $D^{train}$ (the similarity being only computed over the set of problem features) and compute their difference with $C_i$ to create $\Delta D^{train}$: for $C_j = (X_j^{train}, y_j^{train})$ one of the $ntr$ similar cases to $C_i$, add to $\Delta D^{train}$ the case difference $(X_i^{train} - X_j^{train}, y_i^{train} - y_j^{train})$.
3. Fit the neural network to $\Delta D^{train}$ over a number of iterations *epochs*.
4. For each $X_j^{test}$, retrieve $nte$ similar cases $(X_i^{train}, y_i^{train}) \in D^{train}$, and compute the differences $\Delta X_{ij}^{test} = X_j^{test} - X_i^{train}$. Let $\Delta X_j^{test} = \left\{\Delta X_{ij}^{test}\right\}_i$ be the set of these differences.
5. Use the trained neural network to predict the differences in values corresponding to each of the difference feature sets $\Delta X_{ij}^{test} \in \Delta X_j^{test}$, calling the result $\Delta X_{ij}^{pred}$. Compute $y_j^{pred}$ as the average over $i$ of $y_i^{train} + \Delta X_{ij}^{pred}$, with $y_i^{train}$ being the value from the original training set associated with the corresponding similar case to $X_i^{test}$.
6. Compare $y_j^{pred}$ to $y_j^{test}$ to assess the accuracy of the predictions.

Several of those steps (splitting of the dataset, training, prediction) can be implemented in the same way as for the previous process. To facilitate the implementation of steps 2, 4 and 5, we created a python library called $deltaML$.[4]

[1] those features being one-hot encoded.
[2] https://github.com/keras-team/keras
[3] https://github.com/scikit-learn/scikit-learn
[4] https://github.com/mdaquin/deltaML

This library uses the $NearestNeighbour$ function from the *scikitlearn* library as a reliable implementation for steps 2 and 4. It is worth mentioning that this process aligns with the one of CBR in the sense that Step 4 corresponds to the retrieval step of CBR, and Step 5 to the adaptation step.

Our experiments correspond to varying a number of factors, as parameters, in both those processes to see how the training performance of the resulting networks is affected. In all experiments, we use the same topology (same number of hidden layers and same number of neurons in each hidden layer) for the neural network whether it is trained in the usual way (Process 1) or from differences (Process 2). The two parameters shared between the two processes are the size of the training/test sets in percentage of the overall dataset ($test\_size$) and the number of epochs used for training ($epochs$).

In addition, for the difference-based process, two additional parameters are included: The number of similar cases used during training ($ntr$ in Step 2), and the number of similar cases used in prediction ($nte$ in Step 4).

Finally, a variant (originally described in [4]) of the difference-based process is also tested in which, in addition to the differences between feature sets, the original feature sets of the considered case are also included as context. In other words, in Step 2, the created entry in $\Delta D^{train}$ corresponds to $((X_i^{train}, X_i^{train} - X_j^{train}), y_i^{train} - y_j^{train})$ and a similar change is made in Step 4 to include the *context* of the difference (i.e. the original set of features).

## 4 Factorial Study Methodology

The proposed factorial study consists, for each given dataset, in training and testing a neural network with a range of values for each of the parameters/factors described above. To do so, a preliminary phase was carried out where for each of the datasets, we first experimentally identify a network structure (reused for all tests on that dataset) and a set of parameters that obtained good results on a network trained in the usual way. In other words, through trial and error in the application of Process 1 of Section 3, we first identify the kind of neural network to use (number of layers, size of layers, etc.) so to achieve good results when training in the usual way. Once those base parameters are established, we iteratively vary the values of the considered parameters, training and/or testing a neural network model at each iteration.

We apply this factorial analysis on three different processes to compare their performance under different conditions:

**Base:** In this version, we train a neural network in the usual way, using the first process described in the previous section.

**Differences:** In this version, we train a neural network to predict differences in values from differences in features, using the second process described in the previous section.

**Differences+context:** In this version, we train a neural network to predict differences in values from differences in features and the features of the original case, using the variant of the second process used above, as described at the end of the previous section.

All the code and configurations to reproduce those tests and the results obtained are available in the repository hosting the *deltaML* library. In the following, we described the datasets used in the experiments, and the range of values used for each parameter and dataset.

### 4.1 Datasets

Below is a short description of each dataset used in our experiments, including their variants. All those datasets come from shared public libraries, and have been used as downloaded from the links provided. For all datasets, categorical values were one-hot-encoded and the dataset was standardized. Unless explicitly stated, no other modification was made. For each dataset, we also specify the structure of the neural network used in all experiments of that dataset and its variants. For all neural networks, we only used the *ReLu* activation function, and the *mean squared error* as the loss function.

*Used cars:* This dataset[5] contains information about used cars (model, year, fuel type, transmission type, fuel consumption, mileage, tax band) and the price at which they were sold. The goal is to be able to predict the sale price of a car given those features. The neural network model used for this dataset is a sequential, feedfoward network including two hidden layers of sizes 50 and 30 respectively. We tested two variants of this dataset, one for cars of the brand Toyota and one for cars of the brand Vauxhall, since those gave significantly different results.

*Airfoil:* This dataset[6] provides information about the parameters of tests in wind tunnels of airfoil blade sections (frequency, angle of attack, chord length, free-stream velocity, suction side displacement thickness) and the noise emitted (scaled sound pressure level) as a result. The goal is to predict the level of noise from those characteristics. The neural network used for this dataset is a sequential, feedfoward network including two hidden layers of sizes 20 and 10 respectively.

*Students:* This dataset[7] provides information about students in a school in Portugal (demographics, family situation, transport, etc.) and the results of their tests (intermediary and final) in Math and Portuguese. The goal is to predict the results of students at the final tests based on the other characteristics. The neural network used for this dataset is a sequential, feedfoward network including two hidden layers both of size 10. We consider the prediction of final test results in Math and in Portuguese as two separate variants of this dataset.

[5] https://www.kaggle.com/code/najibmozahem/used-cars-neural-network/data
[6] https://www.kaggle.com/datasets/fedesoriano/airfoil-selfnoise-dataset
[7] https://www.kaggle.com/datasets/impapan/student-performance-data-set

*Flights:* This dataset[8] provides information obtained at a given date ($11^{th}$ February 2022) about flights (date and time, duration, origin and destination, airline, number of stops) and the price of a single ticket at that date. The goal is to predict the price of the ticket based on the other characteristics. The neural network used for this dataset is a sequential, feedfoward network including two hidden layers of sizes 50 and 10 respectively. Two variants of this dataset are included: one with the price of tickets in economy class, and one with the price of tickets in business class. To keep the training time reasonable, we reduced the dataset, for both variants, to a randomly selected subset of 15,000 flights (from 206,774 flights in economy and 93,487 flights in business).

### 4.2 Factors and parameters

As mentioned above, the objective of this study is to compare the performance of the neural networks trained in the usual way and from differences under varying conditions, and to see how those factors impact on the performance of those networks. Those factors are represented by the following parameters:

*test_size*: The test size corresponds to the relative amount of data from the original dataset $D$ which is kept for testing, as opposed to training. In other words, higher values for the test size imply smaller amounts of data used in training. It is expected that performance should therefore decrease as *test_size* increases. This parameter is used for both networks trained in the usual way and from differences (with and without context).

*epochs*: The number of epochs corresponds to the number of times the training process will iterate over the training set to fit the network. The need for a high number of epochs, when neural networks are trained in the usual way, highly depends on the task, the structure of the network and the dataset. We therefore fixed the range of values to be tested based on the best results obtained through trial and error (as described at the beginning of Section 4). This parameter is used for both networks trained in the usual way and from differences (with and without context).

*ntr*: The number of similar cases used in training corresponds to the number of retrieved nearest neighbours used during the training phase for each case included in the training set (see Step 2, Section 3). It is worth mentioning that *ntr* can be seen as a multiplier for the size of the training set: for each case in the original set, *ntr* case differences will be included in the difference-based training set. This parameter is only used for networks trained from differences (with and without context).

*nte*: The number of similar cases used in testing corresponds to the number of retrieved nearest neighbours used for prediction (see Step 4, Section 3). For each of them, the difference in values will be predicted and an average difference calculated over the *nte* retrieved cases. This average difference is then used to compute the final prediction. This parameter is only used for networks trained from differences (with and without context).

[8] https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction

Table 1 summarises the range of values used for each of the parameters above when testing using each of the datasets described before. For a given dataset, we used the same ranges of parameter values for all three kinds of trained networks (see Section 4) so to be able to compare their performance under the same conditions. We also used the same ranges of parameter values for any variant of a given dataset.

**Table 1.** Ranges of values for each parameter/factor and each dataset in the format (min, max, increment).

| | test_size | epochs | ntr | nte |
|---|---|---|---|---|
| **Used cars** | (0.05,0.95,0.05) | (1, 20, 1) | (1, 5, 1) | (1, 5, 1) |
| **Airfoil** | (0.05,0.95,0.05) | (2, 402, 10) | (1, 5, 1) | (1, 10, 1) |
| **Students** | (0.05,0.95,0.05) | (1, 121, 5) | (1, 5, 1) | (1, 10, 1) |
| **Flights** | (0.05,0.95,0.05) | (1, 51, 5) | (1, 5, 1) | (1, 10, 1) |

## 5 Results

In the following, we summarise the results obtained based on the factors considered in our experiments. In all cases, we use the $R^2$ score to measure the performance of each of the trained neural networks (Steps 4 and 6 respectively in the two processes described in Section 3). The summary of the results is that the same networks trained from differences on the same datasets are in some circumstances able to outperform the ones trained in the usual way, but the best achievable performance are generally not significantly different. What is different however is that the networks trained from differences appear to require less training than the ones trained in the usual way: They reached close to their best results from significantly less epochs of training. We also show that the numbers of similar cases used during training and prediction affect the results, but that different datasets appear to have different requirements with respect to those numbers. Finally, we show that adding the context (the features of the source item) in addition to the differences in training and prediction does not always have a significant effect on performance, but does in a positive way in some cases.

### 5.1 Performance in relation to test size

Figure 1 shows an example of the evolution of the performance (according to the $R^2$ score) of the neural networks trained in the three different ways according to the size of the test set (i.e. the portion of the dataset that is reserved for testing, as opposed to training) for the economy variant of the Flights dataset. As can be expected, there is for the three versions a general trend downwards (the network

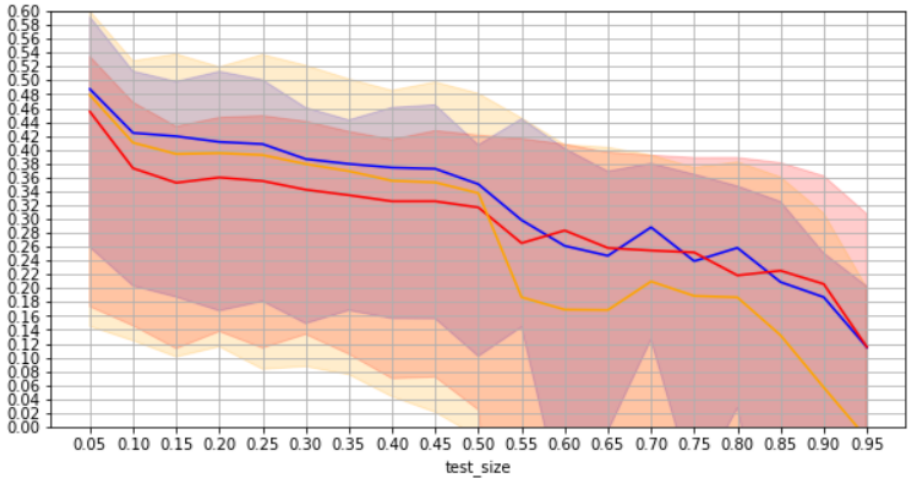


**Fig. 1.** Evolution of performance ($R^2$ score) depending on size of the test set for the Flight/economy dataset. Blue represents Base, red represents Differences and yellow represents Differences+context. For each, the average $R^2$ score across all other parameters than *test_size* is represented by the line, and the minimum and maximum are represented by the borders of the area around it.

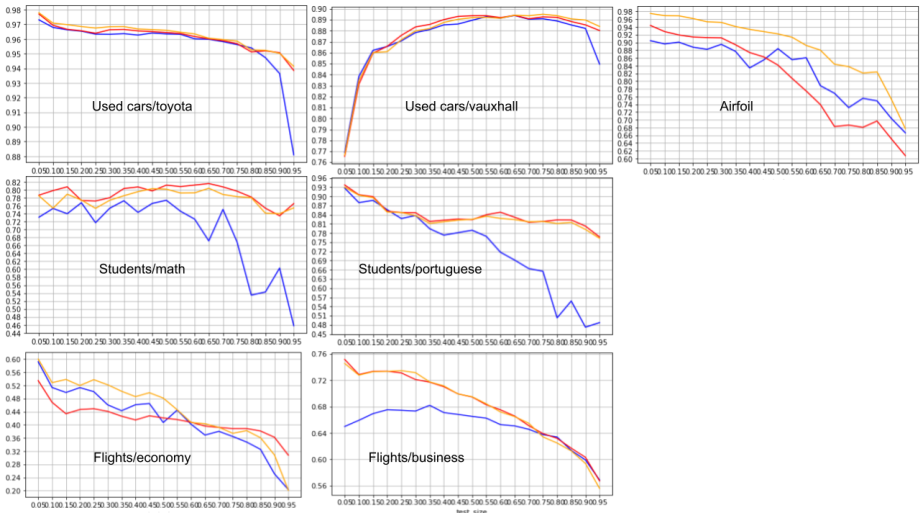


**Fig. 2.** Best performance obtained by Base (blue), Differences (red) and Differences+context (yellow) by test size.

is less accurate with less training data) and they follow a similar shape. Other than that, no general conclusion can be drawn: In some cases Difference+context seems to give better results, in other cases it is Differences and in some others it is Base.

This conclusion on a single example appears to be representative of what can be observed on the whole set of datasets and variants of datasets, as can be seen in Figure 2. Indeed, in all cases, the three curves follow a similar trend. While we can see for both variants of the Students dataset that Differences and Differences+context both obtain better results overall than Base, it is not true to a significant extent for all datasets. Used cars/vauxhall appear to be a special case overall, with lower results for lower test sizes (and therefore higher sizes of the training dataset), but the same trend is visible for the three training methods. In other words, whether learning from differences, with or without context, is more adapted to situations where lower amounts of data are available appears to depend on the dataset and task under consideration.

### 5.2 Performance in relation to the number of epochs

Figure 3 shows the example of the toyota variant of the Used Cars dataset for the evolution of the neural networks performance based on the number of epochs, i.e. the number of times the training process iterates over the dataset. As can be expected, Base, the neural network trained in the usual way, sees its performance increase with the number of epochs. It starts to plateau around 11 epochs and generally reaches its best performance at close to 20 epochs. Also, it can be noticed that at lower numbers of epochs, the performance of Base is more affected by other parameters (namely, the test size) than it is at higher ones (higher spread between minimum and maximum results).

What is however more surprising in this figure is that the trend for both networks trained from differences is significantly different. In both cases, high values of the $R^2$ score are achieved as early as Epoch 1, and they remain high. We can also notice, in this case, that the impact of other parameters ($test\_size$, $ntr$, $nte$) is relatively low both in high and low numbers of epochs.

As can be seen in Figure 4, the same conclusion can be drawn for all the datasets and variants of datasets. In every case, while Base might require anything between 20 and 400 epochs to reach its peak performance, Differences and Differences+context achieve their best results, or close to their best results, from a comparatively low numbers of epochs. This is the most surprising result of this study, as it is hard to explain why, in such a systematic way, the training requirements of a network trained from differences are so significantly lower than for the same network trained in the usual way. We could imagine that this is due to the larger amounts of information used to train the network, since if $ntr$ is greater than 1, several case differences are created for every case in the original dataset. However, as can be seen in Figure 3 looking at the range of results for Differences and Differences+context, the stability of the results with respect to the number of epochs remains even in the worst case. A closer inspection of the

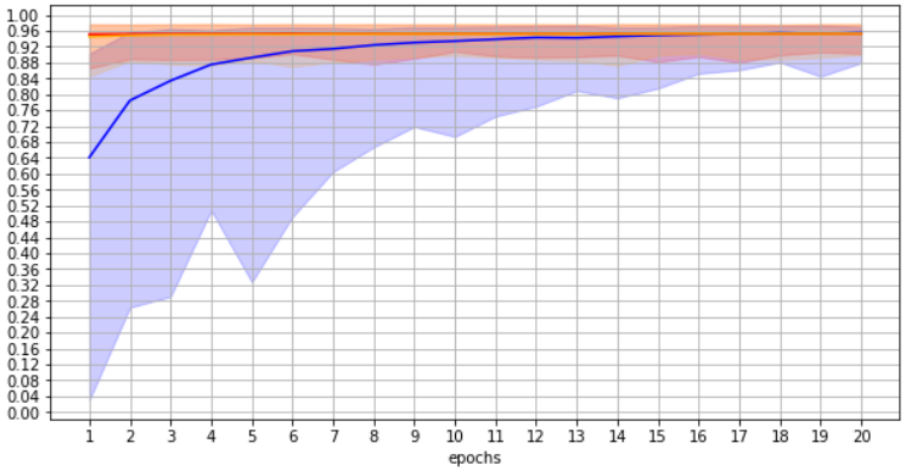


**Fig. 3.** Evolution of performance ($R^2$ score) depending on the number of epochs for the Used cars/toyota dataset. Legend is as per Figure 1.

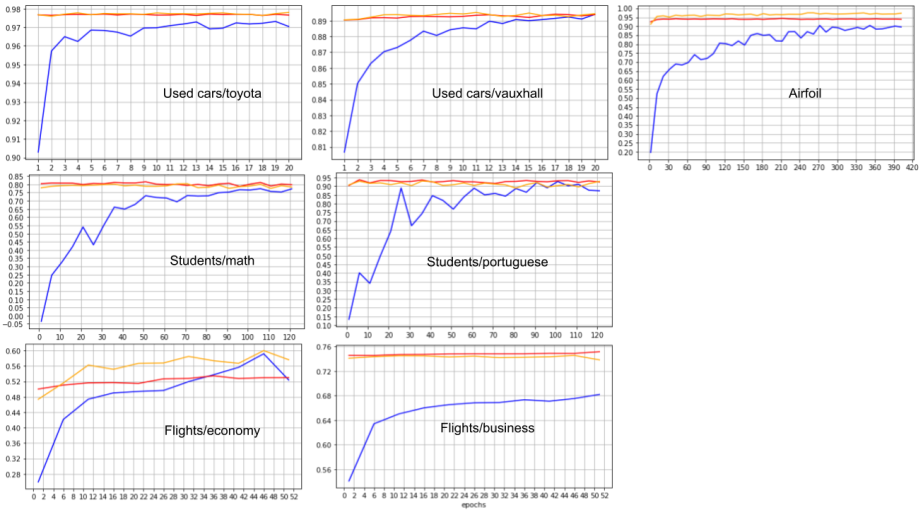


**Fig. 4.** Best performance obtained by Base (blue), Differences (red) and Differences+context (yellow) by number of epochs.

results would indeed reveal that even if *ntr* is 1, the observed phenomena is still visible.

### 5.3 Performance in relation to the number of similar cases used

One of the advantages that the approach by difference has is that multiple case differences can be created by computing the differences in features and results with multiple similar data entries for every item in the original training set. This, in practice, means that the amount of training data can be multiplied without adding any new data. In addition, the ability to aggregate the results from multiple retrieved cases given a target set of features can also help smooth out possible outliers and irregularities in the results at prediction time, obtaining better accuracy. To test this, we varied the number of cases used to create differences in the training set (*ntr*) and the number of items retrieved and aggregated during prediction (*nte*) to see their effect on performance.

Table 2 shows which values of *ntr* and of *nte* obtained the best results on average and as a maximum over all the other parameters for both Differences and Differences+context. As can be seen, there does not seem to be a clear trend in those data. While in some cases, very small numbers of similar cases are required in training and prediction, in others, the best values were obtained with the highest number within the ranges tested. There appears to be a slight trend indicating that Differences+context sometimes require a lower number of similar cases, especially in training, but this difference does not seem significant enough to draw conclusions.

In summary, the best number of similar cases to use, both in training and in prediction, appears to be dependent on the dataset under consideration. Since the role of using those multiple cases can be seen as enabling a greater use of the information about the input space available and as helping smooth out outliers, we can expect those variations to be related to the density of the original dataset, i.e. how many cases tend to share the same area of that data space that are relevant to be used in the same context for prediction.

### 5.4 Peak performance

To get an overview of the main results of the presented factorial study, we look at the minimal requirements in number of epochs, number of similar cases used during training and number of similar cases used during prediction to reach peak performance for each of Base, Differences, and Differences+context. We look at those results for test sizes of 20% and 80% of the original dataset (consistently with [5], corresponding to training sets of 80% and 20% of the original dataset respectively). We consider peak performance to have been reached when the $R^2$ score obtained is within 0.2% of the maximum, to account for the non-significant variations that naturally appear in those scores. The result of this analysis is presented in Table 3.

The first result from this table is that, as noticeable in the previous figures, at least one of the methods using differences often slightly outperforms the base

**Table 2.** Number of cases used to create the difference-based training set (*ntr*) and number of cases retrieved/aggregated for prediction (*nte*) obtaining the best results on average and in the best case.

| | Differences | | | | Differences+context | | | |
|---|---|---|---|---|---|---|---|---|
| | *ntr* | | *nte* | | *ntr* | | *nte* | |
| | mean | best | mean | best | mean | best | mean | best |
| Used Cars/toyota | 4 | 5 | 5 | 4 | 3 | 3 | 5 | 4 |
| Used Cars/vauxhall | 5 | 5 | 5 | 5 | 2 | 3 | 5 | 5 |
| Airfoil | 5 | 5 | 2 | 2 | 5 | 2 | 2 | 2 |
| Students/maths | 4 | 3 | 9 | 10 | 4 | 1 | 10 | 10 |
| Students/portuguese | 5 | 5 | 9 | 10 | 5 | 1 | 8 | 9 |
| Flight/economy | 5 | 5 | 8 | 6 | 3 | 5 | 10 | 6 |
| Flight/business | 5 | 5 | 4 | 2 | 4 | 2 | 5 | 2 |

method where the neural network is trained in the usual way. In the few cases where it does not, the $R^2$ scores actually obtained are very close to the best results obtained with Base.

What can be noticed as well is that adding the context (the features of the source item used to construct differences) to the input variables does not always lead to greater performances. However, if lower, the results of Differences+context are close to the ones of Differences, while if higher, they can be significantly higher (e.g. for Airfoil with both 20% and 80% test sizes). In other words, including the context seems a valid option since it does not generally lead to drastically lower results, while potentially bringing significant improvements.

As already mentioned in the previous section, it is difficult to find a pattern of interest in the number of similar cases used in training and prediction to achieve the best results in Differences and Differences+context. This is true also when comparing the results between the two difference-based methods: sometimes Differences+context requires slightly more similar cases, and sometimes slightly less, but the numbers are always relatively similar.

Finally, the most striking result is, as already discussed in Section 5.2, that in most cases, the methods learning from differences require significantly less training than the Base method. Indeed, the number of epochs required to achieve a score close to the best result is in most cases less than half of the one required by Base. Here too, it is difficult to find a pattern in comparing the required number of epochs between the two difference-based methods: Differences+context sometimes require less, and sometimes more. It is worth reminding the reader in addition that, according to figures 3 and 4, it is not only the case that learning from differences achieves its best results earlier (in number of epochs), but also that those results are more stable: While a slight decrease in number of epochs with Base might result in a significant drop in performance, the $R^2$ score tends to stay within a short range of the best result even when drastically reducing the amount of training carried out.

**Table 3.** Overview of parameters reaching within 0.2% of the best results for Base, Differences and Differences+context with sizes of test sets of 20% and 80% (i.e. sizes of training sets of 80% and 20% respectively). *ep.* corresponds to the number of epochs.

| | Base | | Differences | | | | Diff+context | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $R^2$ | *ep.* | $R^2$ | *ep.* | *ntr* | *nte* | $R^2$ | *ep.* | *ntr* | *nte* |
| **20% test size** | | | | | | | | | | |
| Cars/toyota | 96.5% | 10 | 96.6% | 1 | 3 | 5 | **96.8%** | 6 | 5 | 5 |
| Cars/vauxhall | **86.6%** | 17 | **86.6%** | 16 | 3 | 5 | 86.1% | 3 | 1 | 5 |
| Airfoil | 88.8% | 392 | 91.4% | 152 | 2 | 2 | **96.2%** | 232 | 5 | 10 |
| Students/math | 76.7% | 96 | 77.3% | 6 | 5 | 9 | **77.4%** | 66 | 1 | 10 |
| Students/port. | **85.7%** | 116 | 85.2% | 21 | 4 | 9 | 84.9% | 21 | 3 | 10 |
| Flights/economy | 51.3% | 51 | 44.6% | 46 | 5 | 9 | **52.0%** | 46 | 5 | 9 |
| Flights/business | 67.5% | 46 | **73.4%** | 11 | 5 | 3 | 73.3% | 6 | 5 | 3 |
| **80% test size** | | | | | | | | | | |
| Cars/toyota | **95.4%** | 19 | 95.1% | 1 | 4 | 5 | 95.3% | 4 | 4 | 5 |
| Cars/vauxhall | 88.9% | 20 | 89.2% | 7 | 5 | 5 | **89.4%** | 4 | 4 | 5 |
| Airfoil | 75.5% | 262 | 68.0% | 232 | 5 | 10 | **82.1%** | 382 | 5 | 10 |
| Students/math | 53.5% | 66 | **78.2%** | 16 | 5 | 9 | 78.0% | 41 | 3 | 9 |
| Students/port. | 50.3% | 111 | **82.3%** | 46 | 5 | 8 | 81.2% | 76 | 3 | 8 |
| Flights/economy | 34.7% | 51 | **38.8%** | 21 | 5 | 9 | 38.2% | 11 | 5 | 10 |
| Flights/business | **63.4%** | 46 | 63.2% | 31 | 5 | 6 | 62.5% | 6 | 5 | 10 |

## 6 Conclusion

In this paper, we presented a factorial study comparing the influence of the size of the training set, the number of epochs of training and the number of similar cases used on the performance of neural networks trained in three different ways for regression tasks: The usual way (Base), where values are predicted from the raw input data, from differences (Differences), where differences in values are predicted from differences of pairs of input vectors, and from differences with their context (Differences+context), where differences in values are predicted from differences of pairs of input vectors and the raw feature data.

The main findings from this study are that, from all the seven datasets and variants of datasets tested: 1- the performance of the difference-based methods tend to be comparable, and often slightly higher than the performance of the Base method; 2- there seem to be an advantage in adding the context (in the form of the original set of features) to case differences as it achieves either very similar or better results (as already discussed in [4]); 3- Both methods based on learning from differences in most cases required significantly less epochs of training to reach their peak performance, and arrive within a short range of that best result in just a few epochs. This last point is significant since it implies that those difference-based methods can achieve performances at least as good as the Base method, while being trained for less time. This, however, has to be considered carefully since the retrieval of similar cases and the inclusion of multiple case

differences in both training and prediction also comes with a time overhead. It would therefore be interesting to study in more details the time implications of those aspects and to test more efficient implementations of similarity based retrieval than the one used in the *deltaML* library developed for this study.

In addition, the present study has a number of limitations and would therefore benefit from being further expanded. First, we only considered regression tasks. Since it is not guarantied that the results found here would generalise to neural networks trained for classification tasks (as in [7]), a similar study on such networks would be beneficial. Also, while the datasets selected are varied in size, dimensionality, topics and the results obtained, the neural networks used on them remain similar. In addition to including classification tasks, the inclusion of datasets and tasks requiring different types of networks and a broader range of network size/depth would help confirm the results obtained. Finally, not all of the factors influencing the performance of a neural network have been considered here. In particular, we deliberately did not vary the structure of the network in terms of number of layers, neurons per layers, activation functions and loss function. It could be possible, however, that a different structure be more suitable for learning from differences than for learning in the usual way, leading to possibly even better results.

## References

1. Craw, S., Wiratunga, N., Rowe, R.C.: Learning adaptation knowledge to improve case-based reasoning. Artificial intelligence **170**(16-17) (2006) 1175–1192
2. Hanney, K., Keane, M.: Learning adaptation rules from a case-base. In Springer, ed.: Third European Workshop on Case-Based Reasoning. (1996) 179–192
3. Jalali, V., Leake, D., Forouzandehmehr, N.: Learning and applying case adaptation rules for classification: An ensemble approach. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. (2017) 4874–4878
4. Leake, D., Ye, X., Crandall, D.J.: Supporting case-based reasoning with neural networks: An illustration for case adaptation. In: AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering. Volume 2. (2021)
5. Liao, C.K., A. Liu, Y.C.: A machine learning approach to case adaptation. In Springer, ed.: IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). (2018) 106–109
6. Lieber, J., Nauer, E.: Adaptation knowledge discovery using positive and negative cases. In: ICCBR 2021 - 29th International Conference on Case-Based Reasoning, Salamanca (Virtual), Spain (September 2021)
7. Ye, X., Leake, D., Jalali, V., Crandall, D.: Learning adaptations for case-based classification: A neural network approach. In: Case-Based Reasoning Research and Development: 29th International Conference, ICCBR 2021. (09 2021) 279–293
8. Ye, X., Zhao, Z., Leake, D., Wang, X., Crandall, D.: Applying the case difference heuristic to learn adaptations from deep network features. In: IJCAI 2021 workshop on Deep Learning, Case-Based Reasoning, and AutoML: Present and Future Synergies, arXiv (2021)