Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

Evaluation of Robustness and Performance of Early Stopping Rules with Multi Layer Perceptrons

Aleksander Lodwich, Yves Rangoni, Thomas Breuel

Abstract— **In this paper, we evaluate different Early Stopping Rules (ESR) and their combinations for stopping the training of Multi Layer Perceptrons (MLP) using the stochastic gradient descent, also known as online error backpropagation, before reaching a predefined maximum number of epochs. We focused our evaluation to classification tasks, as most of the works use MLP for classification instead of regression. Early stopping is important for two reasons. On one hand it prevents overfitting and on the other hand it can dramatically reduce the training time. Today, there exists an increasing amount of applications involving unsupervised and automatic training like i.e. in ensemble learning, where automatic stopping rules are necessary for keeping training time low. Current literature is not so specific about endorsing which rule to use, when to use it or what its robustness is. Therefore this issue is revisited in this paper. We tested on PROBEN1, a collection of UCI databases and the MNIST.**

I. Introduction

MULTI LAYER PERCEPTRONS (MLP) trained with Online Back Propagation (OnBP) are very popular tools for classification partly due to their usually fast recall, good performance, the simplicity of the idea and their easy implementation. Caruana's [1] last large scale comparison among supervised learning algorithms revealed that a MLP is still a competitive and one of the best non calibrated methods. *"Of the earlier learning methods, feed-forward neural nets have the best performance and are competitive with some of the newer methods, particularly if models will not be calibrated after training"*[1]

Beyond their simple structure and their simple training method, the MLP+OnBP combination has the advantage that it can easily deal with extremely large and non sparse datasets. The online behaviour of the MLP and OnBP makes it one of popular online learning methods (detailed introduction to this topic given in [2]) where labels can be obtained (i.e. from the old network state).

There exists an increasing number of new applications where Artificial Neural Networks (ANN) such as the MLP's are used as weak learners in a boosting or bagging context [4] or are used in automatically adapting black box systems [2]. These are systems where no user helps adapt any parameters. In other cases, evolutionary algorithms are used in order to optimize the architecture of an MLP [5]. This involves many training runs where unsupervised training control is required. Part of this control, and its the main motivation for the here presented investigations, is to make the choice when to stop the training in such a way that computational effort is minimized but not at the expense of good generalization.

This means, that the question when to stop is important not only because of speed considerations but also because neural networks cannot be trained forever without overfitting to the training dataset (loosing generality). Early stopping has proven to be a quite effective method of avoiding overfitting. Prechelt [7] summarizes Finno et al. [6]: *"Early stopping is widely used because it is simple to understand and implement and has been reported to be superior to regularization methods in many cases"*.

A brief overview over the key problems is: If an Early Stopping Rule (ESR) stops too late, than we miss out faster training and eventually loose generality. If it stops too early, then we miss out on the accuracy of the predictions. If the rule hardly ever fires, we gain nothing and need to prune, regularize or use cross-validation anyway.

The state-of-the-art method is a cross-validated stop where a network is recorded and trained for a long time and then gets cross-validated at recorded epochs [1]. Best stored version is taken. The disadvantage is that the network needs to be stored at least one time and that the cross-validation needs to be performed in many epochs. This method clearly aims at getting the best solution. In its pure form, it will require training of the network up to the predefined maximum number of epochs. If this number is large then this can result in an excessively long training phase. In that sense this method cannot be considered a "real" ESR.

A traditional approach to early stopping [3] is to stop training manually by comparing the progress of the Mean Squared Error (MSE) for the training set and for the validation set. One approach is to stop when the training MSE does not improve, another one is to stop when the MSE on the validation dataset begins to wander up. In other cases it seems plausible to stop at the epoch where the dataset size corrected MSE curves cross.

The straightforward ideas collected with manual stopping have led to automated methods using different rules. In this paper, we aim to produce a fair evaluation of major published rules, in a clear and solid framework, and find a clear answer for the question whether there exists a good ESR that can deliver a good balance between training efficiency (stopping at the right place), the quality of the resulting neural networks, and the frequent production of early stops (robustness). During the investigation of already published work it appeared, that the presented evaluation methods were either not comparable or the measured numbers were too weak to give a clear instruction which method to use.

Additionally, to the evaluation of individual ESR, we will also investigate whether a combination of two or three ESR can produce better results and draw an overview of which ESR should be combined.

II. Description of the Stopping Rules

In this section we want to revisit stopping rules of which we are aware of. We differentiate the following rules into Primary Rules (PR) and Secondary Rules (SR).

A. Primary Rules

Primary rules are rules known to become true at some point in time during a training and are a necessary frame for evaluation of rules which stop cannot be clearly predicted. Application of primary rules is therefore mandatory.

The only primary rule we use is the Maximum Number of Epochs. We set this number to 1000 epochs as further increase did not improve the ratio of early stopped runs to all runs. The histogram in Fig. 1 shows that the number of early stops beyond 1050 epochs is marginal. We experimented with up to 3000 epochs in order to estimate the number of epochs to run the full experiment and we observed that given the PROBEN1 [12] dataset three quarters of successfully early stopped runs are being achieved *independently* of the further limit extensions. It seems that there must be a saturation point, where more than a specific number of runs cannot be covered. Hence we think that the limit of 1000 epochs is fully justified for this kind of experiment. Note that for large datasets, with few classes, like the MNIST, even few dozen of epochs are largely sufficient.

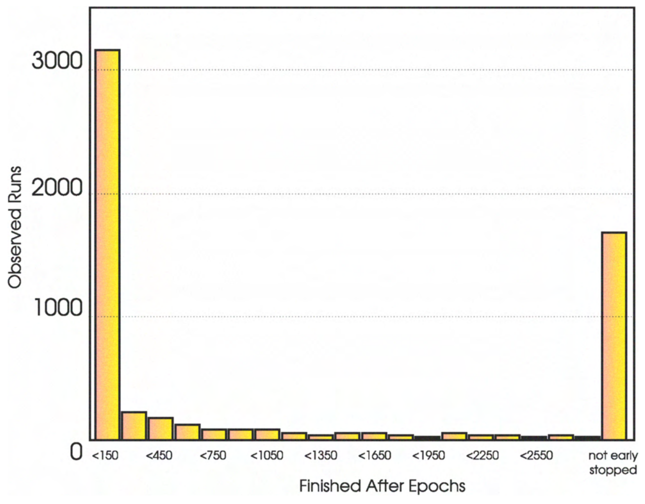

Fig. 1. Most of the early stops happen before 1000 epochs: more than 70%. In many fail cases, increasing the number of epochs, even by several thousands, will not allow the rule to stop anyway.

B. Secondary Rules

Secondary Rules are rules that cannot be guaranteed to fire sometime but carry the potential to improve average accuracy and training speed.

We tested six different families of rules from which three had been evaluated before in [7] or [8] for RPROP instead of the OnBP. All the rules make use of the computed error. Rules harnessing accuracy information were not tested because this information is rarely available.

1) Loss of Generality (GL): The Loss of Generality is an automatic rule adapted from manual stopping procedure. The training is stopped when the error of the validation dataset rises above a specific threshold which is relative to the minimal error observed thus far [8].

The optimal error is defined as:

$$E_{opt}(t) = \min_{t' \leq t} E_{valid}(t')$$

Then the loss of generality is defined as:

$$GL(t) = 100 \cdot \left(\frac{E_{valid}(t)}{E_{opt}(t)} \right)$$

The rule GL_α is defined as:

$$GL_\alpha : GL(t) > \alpha$$

The tested α were 1, 2, 3, 4 and 5.

2) Filtered Loss of Generality (GSC): Many learning curves can be quite noisy and it seems plausible to smooth them in order to get only relevant information. For this paper, we added a special version of GL to our experiment that applies a Gauss filter to the curve. Filterwidth is 25 epochs and σ is 3. These values were set after manual experimenting.

3) Low Progress (LP): The low progress rule (LP) fires when the improvements on the training error stall. It is a common rule, even described by Prechelt, but it has not been evaluated by him. In order to be comparable, we reuse the P_k function from him.

$$P_k = 1000 \cdot \left(\frac{\sum_{t'=t-k+1}^{t} E_{train}(t')}{k \cdot \min_{t'=t-k+1}^{t} E_{train}(t')} \right)$$

This basically says to look at how much change has there been in the past k steps. We use a proposed k of 5 [8]. Low Progress is then defined as:

$$LP_\alpha : P_5(t) < \alpha$$

The tested α were 1, 2, 3, 4 and 5.

4) Steady-State Stop Training Trigger (SSSTT): This method was first published in [9]. The underlying concept is to stop training when the improvement in prediction is insignificant relative to the variability in the residuals computed from the training on a random 20% to 30% small subset of the training dataset in each epoch. When noise exceeds learning effects then stop training. The paper has done a successful proof of concept on three datasets from the PROBEN1 [12] database. It would be useful to evaluate this method on more datasets but it has four parameters to set and therefore was not compatible with our experimentation scheme. We consider four parameters (even if the author claims to have found a robust general setting) to set as a "too much" for the simple job an ESR has to do. Therefore we introduced a similar rule as replacement: NR_α.

5) High Noise Ratio (NR): Motivated by the SSSTT rule we introduce another rule, that comes from the observation that in some cases the training curves E_{tr} get noisier at the end indicating the attempt of the network to overfit. A rule for stopping before noise grows too high is the High Noise Ratio rule (NR). This rule is an indicator of energy stored in the second derivative.

$$NR_k(t) = \frac{\sum_{i=1}^{k} E_{tr}(t-i) - 2 \cdot E_{tr}(t-i-1) + E_{tr}(t-i-2)}{\sum_{j=1}^{k} E_{tr}(t-j)}$$

Stopping occurs when NR with a k of 20 exceeds either 5, 10, 15, 20 or 25. The k value was set to the best value from experience.

$$NR_\alpha : NR_{20}(t) > \alpha$$

6) Generality to Progress Ratio (PQ): As discussed in [8] it seems sometimes favourable to accept a higher loss in generality when there is a higher progress on the training set in the hope that there will be improvements on the validation set later. This is formulated as:

$$PQ_\alpha : \frac{GL(t)}{P_5(t)} > \alpha$$

Used α were 1, 2, 3, 4 and 5.

7) Consecutive Loss in Generality (UP): While GL is a global method that takes the minimum of all of the observed samples, Prechelt also investigated a local version of it with the name UP. It is not clear what UP stands for (probably just for going up) but it is defined in [8] as:

$$UP_k : \underbrace{E_{valid}(t-k) < E_{valid}(t-k+1) < ... < E_{valid}(t)}_{k}$$

Used k were 2, 4, 8, 16 and 32.

C. Combination of Rules

It can be assumed that different ESR will have individual strengths and weaknesses. Since the computation of the ESR is not very expensive, many rules can be applied at once in combination:

$$ESR_{combined} = ESR_1 \vee ESR_2 \vee ESR_3 \vee \cdots \vee ESR_n$$

Generally, it is not intuitive how to combine ESR in such a way that improves three parameters: fail rate, quality and speed. We evaluated permuted combinations of size 2 and 3. The picked ESR were always taken from different families. Every family was represented by five different α. Thus the total number of investigated rules was 30. In a combined rule, performance values are to be recorded according to the first firing rule out of the group.

III. Experimental Setup

A. Environment

The experimentation environment was Python 2.5.1 with NumPy 1.0.3. The MLP was implemented using NumPy arrays. Tools used were coming from the PaREn project [10].

B. Parameters

We investigated six different families of early stopping rules (described above), with five parameters each on ten different datasets from the PROBEN1 [12] database provided with the PaREnDatabases module: cancer, diabetes, gene, glass, heart, horse, iris, soybean, thyroid and wine. All of them are coming from the UCI repository [17].

We were only interested in the classification task and not in the regression tasks because we wanted to investigate the effects on the accuracy. We decided against encoder problems [14] as they can be considered unrealistically simple problems for stopping rules that can skew the results. Instead, we added more application relevant evaluations on the MNIST 10k and the 60k [13] dataset.

For every dataset a fully connected feedforward MLP was trained with a single hidden layer composed of 16, 32, 64, 128 and 256 hidden units. The network architecture did not involve shortcut connections. From practical projects we know, that three layer networks without shortcuts are still being used extensively if not most of the time. We considered these values large enough for the problems we used, and the values allowed for overfitting in many cases. It was not attempted to optimize the number of weights when compared to the number of samples from the dataset as we wanted to give a general proposition which method to use in uninformed training environments, where little knowledge about the relationship between dataset, network architecture and parameter setup is used to control the training.

The training of the neural network has been performed with four different learning rates $\{0.1, 0.3, 0.5, 0.7\}$ which can be considered as adapted to classification tasks and problems without hard constrains around the optimum. We believe this is true for all of the tested datasets. In case of regression learning rates under 0.1 are common.

The activation function of all the neurons is the logistic function. Input and output values were not normalized.

C. Experimental Procedure

All runs were trained with the online backpropagation (OnBP) [11] for 1000 epochs, as explained in Section II.

The training was repeated eight times in order to have results for different initial weights. Every time, we have had specified a specific seed for the random generator, so that a reproduction of the results is possible. A small initial experiment has shown that setting a seed was sufficient to reproduce training results and final weights.

As usually, we chose to split the original datasets into 3 subsets: training, validation and testing, with the corresponding proportions of 70%, 20% and 10%. Random permutations of the original data are used to feed the subsets. Here also, due to the same seed, the splits are reproducible. After this step, we trained and recorded the MSE and the accuracy for each split. At the end, we produced 1720 graphs, like Fig. 4, with the three curves each with values for 1000 epochs. We refer to these graphs as "runs".

Then, we applied the ESR, that created at total of 51600 records containing information about which database and which network parameters were used and what stopping result was obtained. These records include the measures discussed in the next chapter.

We decided to combine pairs and triples of ESR instances coming from different families. The purpose is to investigate whether a group of rules can yield better performance than a single rule. Here, combining means stopping when the first rule in the group fires.

Sometimes, a failure occurs when a stopping rule does not stop before the number of 1000 epochs. A combination fails when all of the ESR fail. A failure rate is the ratio of failures divided by the number of runs. We decided to remove combinations with failure rates over 25%. At the end there were 2850 combinations under test.

From the 51600 incident records, we created several tables of the kind *dataset vs. ESR* for different investigated parameters. An example in Table I is shown.

TABLE I

POWER OF 1.05 THAT DESCRIBES THE DEVIATION FROM OPTIMAL ACCURACY ACHIEVABLE WITHIN 1000 EPOCHS

Std	cancer	diabetes	gene	glass	heart	horse	iris	mnist10k	mnist60k	soybean	thyroid	wine
GL1	9.18	7.07	5.46	4.39	6.82	4.68	17.53	10.83	10.84	13.02	7.63	5.066
GL2	9.19	7.08	5.27	4.38	6.68	4.98	17.46	10.68	10.94	13.17	7.55	5.079
GL3	9.20	7.08	5.25	4.41	6.69	5.30	17.46	10.00	11.01	13.15	7.52	5.001
GL4	9.22	7.10	4.81	4.31	6.64	4.74	17.76	9.99	10.93	13.28	7.50	5.011
GL5	9.22	7.19	4.44	4.29	6.43	4.69	17.76	9.72	10.75	13.17	7.49	5.10
GSC1	8.61	5.91	3.16	4.02	6.68	4.35	18.86	8.55	7.93	11.67	7.39	4.88
GSC2	8.61	5.87	2.64	3.97	6.72	4.33	18.89	8.28	7.87	11.36	7.40	4.97
GSC3	8.61	5.97	1.88	4.09	6.81	4.40	18.91	8.34	7.88	11.38	7.39	4.90
GSC4	8.65	5.98	1.93	4.06	6.86	4.40	18.82	8.29	8.21	11.41	7.37	4.72
GSC5	8.65	5.77	1.77	4.14	6.78	4.35	18.74	8.36	8.31	11.62	7.36	4.90
LP1	8.80	6.02	3.35	4.61	6.42	4.47	19.16	2.66	1.34	9.43	9.50	5.18
LP2	8.83	6.02	3.36	4.62	6.30	4.47	19.16	2.66	1.34	9.69	9.67	4.84
LP3	8.83	6.02	3.36	4.55	6.23	4.51	19.16	2.64	1.45	9.68	9.67	4.86
LP4	8.78	6.02	3.35	4.54	6.52	4.51	19.17	2.65	1.52	9.65	9.76	4.86
LP5	8.95	6.02	3.33	4.57	6.50	4.56	19.17	2.71	1.52	9.92	9.76	4.80
NR1	8.60	6.23	3.06	4.08	7.18	4.26	18.41	7.29	7.05	11.06	7.34	5.00
NR10	8.65	6.08	2.22	4.08	7.07	4.03	18.42	7.29	7.05	11.19	7.34	5.00
NR15	8.65	6.08	2.06	4.08	7.07	3.94	18.55	7.29	7.05	11.20	7.34	5.00
NR20	9.25	6.20	1.99	4.08	7.09	3.96	18.52	7.29	7.00	11.27	7.36	4.96
NR5	8.60	6.19	2.49	4.07	7.11	4.01	18.46	7.29	7.05	11.09	7.34	5.00
PQ1	9.34	7.03	3.38	4.59	6.43	4.66	18.81	7.65	8.61	10.69	7.96	4.39
PQ2	9.22	6.64	3.26	4.29	6.49	4.52	19.09	5.81	7.31	9.91	8.61	4.56
PQ3	8.87	6.59	3.21	4.40	6.59	4.57	19.12	4.25	2.00	9.85	9.31	4.52
PQ4	9.01	6.51	3.23	4.34	6.20	4.45	18.87	3.78	1.55	9.83	9.86	4.42
PQ5	9.06	6.49	3.23	4.36	6.57	4.60	18.92	3.82	1.58	9.86	10.02	4.68

D. Measuring

Prechelt used three measures for the evaluation of his experiments with RPROP [8] which are Time (here we use Speed Up), Efficiency (here speed axis) and Effectiveness (here quality axis). The problem of Prechelt's measures is that 1) they are expressed as ratio to two other rules within the set and 2) they draw averages on many runs and different datasets first. The result of this choice is that the numbers obtained are very close to each other and suggest that there is no practical difference between the methods. Beyond that it appears to us that Prechelt has drawn means on ratios which are Efficiency and Effectiveness. If that was true, then the numbers were invalid, because ratio-scales cannot be treated like interval-scales. Instead, ratios can be mapped onto an interval-scale by applying logarithms to them [15]. Since we are expressing our ratios as exponents the logarithm-mean is equivalent to the geometric mean of ratios [16].

Our approach to compute Efficiency (expresses speed) and Effectiveness (expresses quality) is very similar to Prechelt's approach but we avoid drawing means on these ratios and use their logarithms instead.

Instead of measuring Time (Prechelt), we quantify how much faster we can be when compared with a full training (Speed Up). Speed Up is a quotient based on the stopped epoch s_e. It is computed as $SU = \frac{s_e}{1000}$. The process of measuring efficiency and effectiveness is illustrated in Fig. 2.

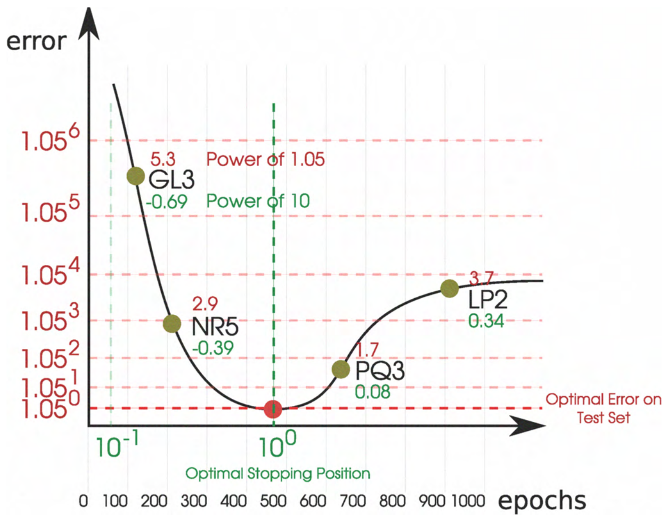

Fig. 2. Speed and Quality are measures according to optimal value.

We express Efficiency as $log_{10}\frac{t_{stop}}{t_{opt}}$ and Effectiveness as $log_{1.05}\frac{M_e(t_{stop})}{M_e(t_{opt})}$ where M_e is a measure of effectiveness. This measure is MSE and accuracy accordingly. The value for t_{opt} is different for MSE and accuracy since these are two different curves and since we are interested in minimum MSE and maximum accuracy. For effectiveness (accuracy/error) we have chosen 1.05 as the base. Theoretically, any other base is equally well suited for the task but this choice allows us to get well readable numbers. For Efficiency (Speed) we have chosen the base of 10 since it is often used for expressing size relationships for natural concepts.

Generally, values closer to zero are preferable.

IV. EXPERIMENTAL RESULTS

The result of the experiment is a file with 51600 records with information about:

- BestPosAccuracy - Epoch number where the test set yielded the highest accuracy. In case of ambiguities the earliest value is taken.
- BestPosError - Epoch number where the test set yielded the lowest MSE. In case of ambiguities the earliest value is taken.
- DB - Name of dataset this value was achieved on
- Early - Bit flag about whether the ESR has fired.
- EarlyOrLateForAccuracy - Exponent to the base of 10 expressing how much earlier or later the rule stopped comapared to the highest accuracy value.
- EarlyOrLateForError - Exponent to the base of 10 expressing how much earlier or later the rule stopped comapared to the lowest MSE value.
- FinishedAfterEpochs - Epoch number where the ESR fired. Possible values 0-999

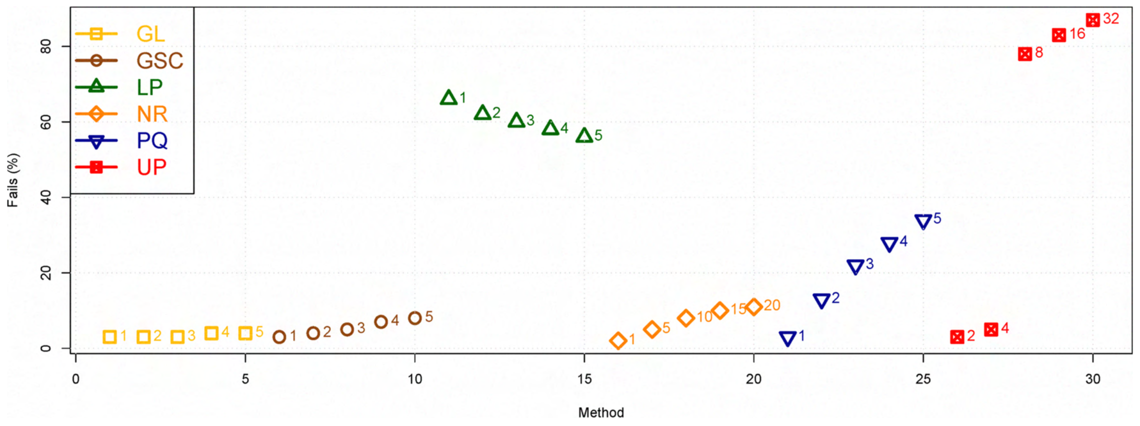

Fig. 3. Rate of failed runs for each method

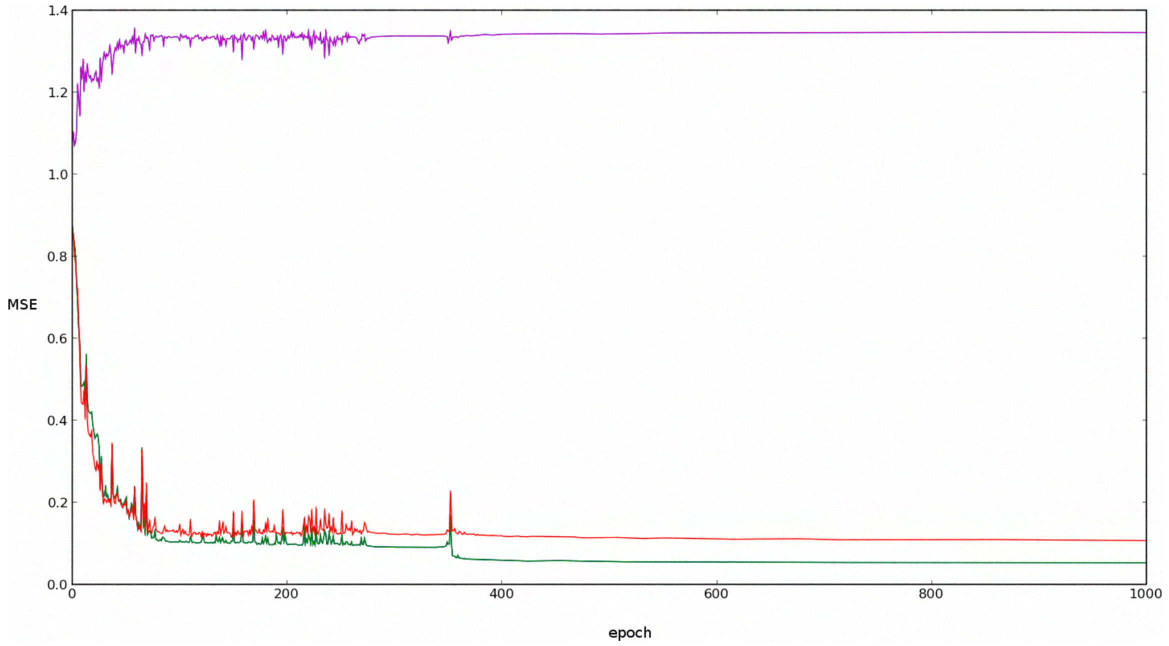

Fig. 4. The performance relationship between training, validation and the test set can be quite counterintuitive

- StoppingRule - String describing one of the 30 different ESR instances.
- $|Error|$ - Integer exponent to the base of 1.05 expressing how much worse the obtained MSE was when compared to the lowest value found using argmin.
- $|Inaccuracy|$ - Integer exponent to the base of 1.05 expressing how much worse the obtained inaccuracy (1-accuracy) was when compared to the highest value found using argmin (=argmax(accuracy)).

In this section we present five graphs which we think clearly express the differences in performance of the ESR. For the purpose of readability the logarithmic values of the measures were expanded to linear scale. The quality axis tells a factor between 1 and 2. A factor value of 1.5 i.e. means that this method produces on average a 50% higher error (or smaller accuracy) than what could be achieved by the cross-validated method as used by Caruana in [1]. The speed axis is also linear. A value of 1 means that an ESR stops at exactly the right place, whereas a value of 2 means, that the ESR lets the training run twice a long as necessary.

In Fig. 5 and Fig. 6, it can be observed that the average performance of the stopping rules is roughly aligned with a hyperbolic function. This is true for both error and accuracy. In Fig. 5, UP_2 is performing 83% worse than what is optimally achievable and runs roughly half as long as required for an optimal stop. Thus this is an example of stopping too early. The LP_1 rule stops ten times later but achieves a better result on average (53%).

Please note, that the values are obtained on a test set that was not involved in the stopping process in any way. Therefore, the values obtained express the generalization abilities of the MLP's. In Fig. 7 and Fig. 8, the same type of diagram is presented but the datapoints are based only on runs that have finished early. While Fig. 5 and Fig. 6

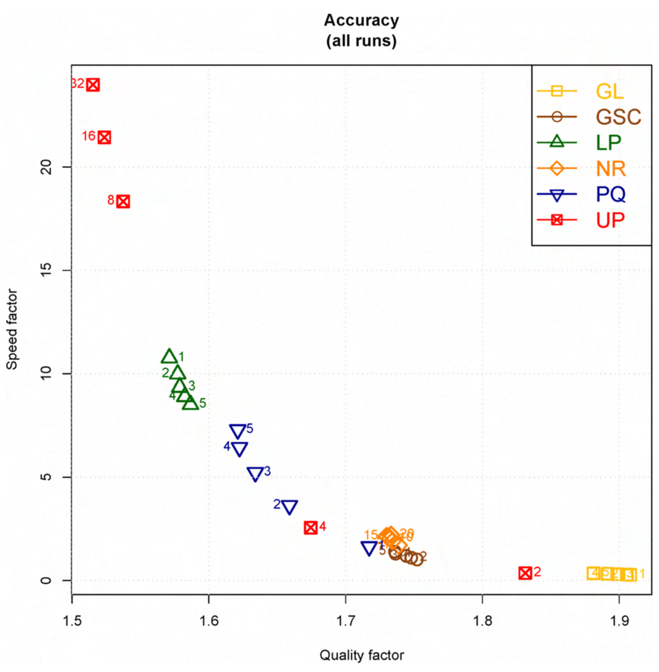

Fig. 5. Relationship between speed and accuracy based on all runs

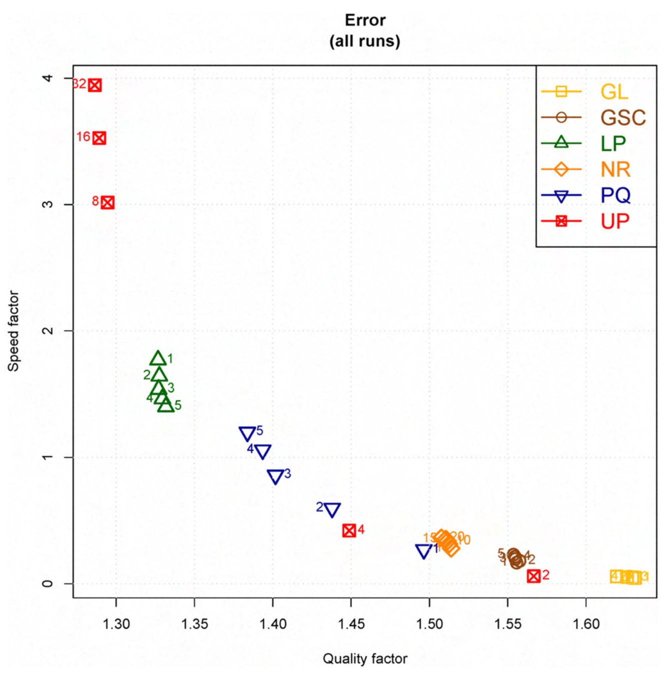

Fig. 6. Relationship between speed and MSE based on all runs

can be used to understand the behaviour of the rules under absolutely no background information, Fig. 7 and Fig. 8 are diagrams which inform about the performance of the ESR when the firing of an ESR can be expected almost for sure.

Table II presents additional information about individual ESR's speed up, because the figures only contain information relative to an optimal stop.

TABLE II
AVERAGE SPEED IMPROVEMENT WHEN COMPARED TO CROSS-VALIDATION

ESR	Speed Up	ESR	Speed Up
GL1	22.70x	NR15	6.00x
UP2	22.40x	UP4	5.80x
GL2	20.60x	NR20	5.70x
GL3	19.20x	PQ2	3.70x
GL4	16.20x	PQ3	2.70x
GL5	14.40x	PQ4	2.30x
GSC1	13.40x	PQ5	2.00x
NR1	11.50x	LP5	1.60x
GSC2	11.20x	LP4	1.50x
GSC3	9.70x	LP3	1.50x
PQ1	8.20x	LP2	1.50x
NR5	8.20x	LP1	1.40x
GSC4	8.00x	UP8	1.20x
GSC5	7.00x	UP16	1.10x
NR10	6.80x	UP32	1.10x

In Table III, we present the 25 best combined rules (tuples & triples) that were obtained under the assumption that failure rates over 25% cannot be tolerated. If tolerated, then the best combination is UP32 and LP1. This combination has achieved a quality value of $q = 1.57$ but had a failure rate of 66%. These values are meant for accuracy and all runs. The quality value q means that if x was the optimal accuracy then the expected performance would be $1-(1-x)\cdot q$. This value is not a lower and not an upper bound. The remaining combinations were sorted by accuracy. According to this table the best tuple combination is PQ3 and UP8 and the best triple combination PQ3, UP32 and LP1. The quality between tuple and triple is roughly the same (1.63 vs. 1.64) but the efficiency of the triple is better (5.99 vs. 5.16). The average speed up is (2.72 vs. 2.80) respectively.

V. Discussion

The diagram numbers convey the specific behaviour of different ESR with different parameters α. For each of the 30 ESR instances we had 160 records to compute the numbers from, except for MNIST [13], where we had 120. The utilized measures deliver a meaningful performance estimation in terms of how efficient and how effective an ESR is.

Unsurprisingly, none of the methods delivers results comparable to cross-validation, but it seems that general statements about the average deviation can be made. First observation is that in absence of any prior knowledge, training using lazy rules like LP will more likely produce better results. Improvements in accuracy or MSE can only be made at great expenses in efficiency. A large efficiency scale (0 to 20) is standing against a small possible range of improvement (for accuracy it is roughly 50%).

It can be observed that ESR stopping roughly at the right point do not necessarily deliver the best results. This is probably due to the high dynamics in the curves around that

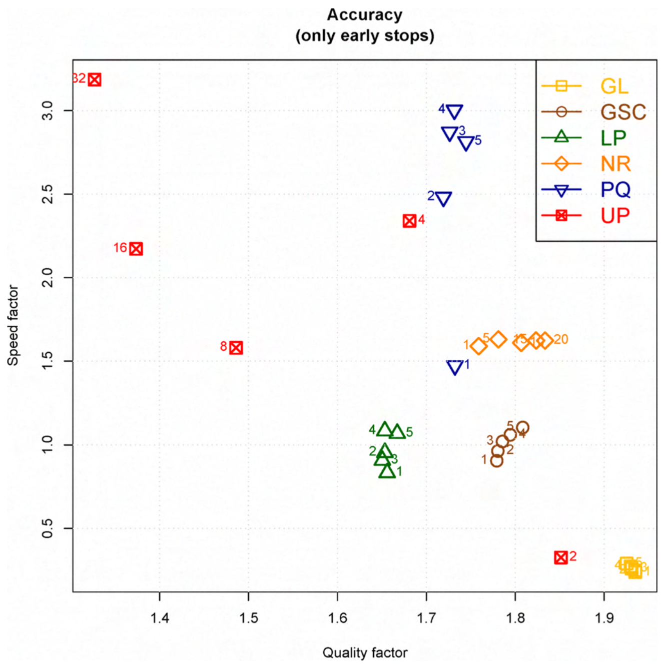

Fig. 7. Relationship between speed and accuracy for early stopped runs

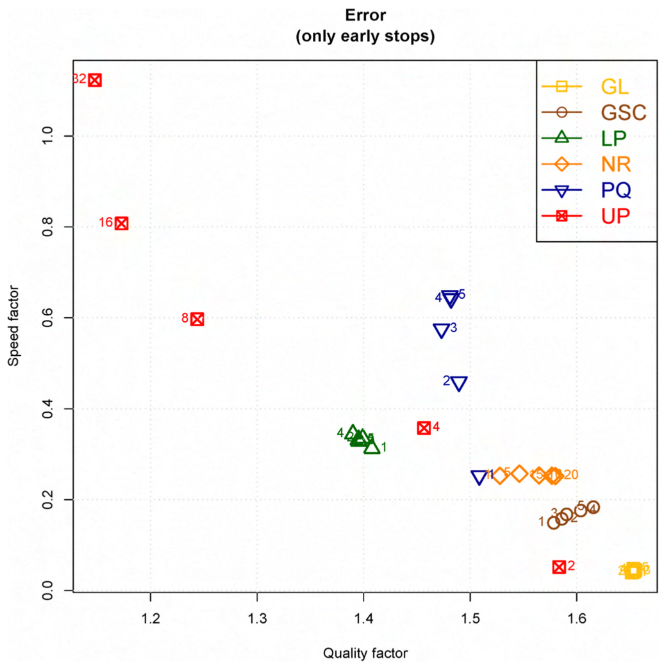

Fig. 8. Relationship between speed and MSE for early stopped runs

optimum, where a small off results in a severe performance degradation. Additionally, the area of optimal performance on the test set can be very far away from the optimal one in the training or the validation set as can be seen in Fig. 4 in extreme example.

If one could assume that the combination of network size and problem (dataset) should or must expose specific curve behaviour then a closer look at the diagrams in Fig. 7 and Fig. 8 can be useful. In both diagrams the UP rule with an α of 32 delivers the best results. The efficiency of the method is much worse than that of the PQ rule and there is hardly any improvement to the training speed when compared with crossvalidation. This means that this rule fires rarely but when it fires it is a strong indicator that training should be stopped. Using PQ instead of UP delivers worse results but reduces average training time by a factor around 2.5.

The fastest methods are the GL rules. These rules mimic a conservative approach to stop training when the MSE on a validation set start to outgrow a specific threshold controlled by the minimum found hitherto. The GL style rules (includes GSC) are the ESR to stop most of the time before reaching 1000 epochs and can therefore be considered very robust rules but the speed advantage is paid with the low quality of the obtained networks. GL style rules predominantly occupy the right side of the diagrams. The absolute speed improvements are large for this family of rules. An improvement at least seven times and maximum 22 times for an experiment using these rules can be expected.

From the diagrams it becomes clear that smoothing over a curve is a good idea. The fail rate of GSC lies hardly above that of GL but improves average quality by roughly 10% on error and roughly 15% on accuracy.

If some further fail rate increases can be tolerated then a noise-ratio based rule should be preferred, because the obtained network quality is generally better than with GSC but not as good as with PQ or LP.

In cases where the assumption of stalling progress can be made, LP is truly the the best method, stopping roughly at the right place and giving best results. That rule is making a quite specific assumption that failed in roughly 50% to 60% of the runs depending on the α. A similar destiny is for the UP rules being very specific, delivering best net performance on the test set but otherwise almost always failing.

Since PQ is a mixture of LP and GL, it is no wonder it resides between them. Depending on the α, it is possible to interpolate the result space between the two rule sets. The PQ family of rules seems to have a good balance between efficiency (speed), effectiveness (quality), fail rate, and global speed improvement. This rule has the highest variance depending on the α (compare with Fig. 3) meaning that choosing α should be done wisely. Choosing an α of 2 will deliver a low fail rate around 10%, a quality around 50% above minimum test set error and inaccuracy (1-accuracy) worse about 70%. These values are to be understood in context to PROBEN1 [12] and MNIST [13] datasets. Practical meaning of these values would be: Let us say the best stopping point revealed an error of 0.25 and an accuracy of 90%, then the expected limits say that with PQ we should achieve results around 0.375 on the error and accuracy around $100-((100-90)\cdot 1.7) = 83\%$. This family of rules yields improvements in training time between two and eight times. Lower values for α yield better speed up.

Using combinations of ESR is possible and yields better results than rules applied in isolation. Almost all the com-

TABLE III

LIST OF 25 FIRST BEST ESR COMBINATIONS. THE *Quality* COLUMN INDICATES HOW MUCH WORSE THE AVERAGE CASE IS WHEN COMPARED TO THE OPTIMUM RESULT AND THE *Speed* COLUMN INDICATES HOW MUCH FASTER A TRAINING WITH THAT RULE IS WHEN COMPARED TO A *cross-validated* STOP WHICH WOULD ALWAYS REQUIRE TO COMPUTE 1000 EPOCHS.

ESR1	ESR2	ESR3	Quality	Speed	Failures
PQ3	UP8		1.63	5.99	21.38
PQ3	UP16		1.63	6.2	21.50
PQ3	UP32		1.63	6.3	21.62
PQ3	UP32	LP1	1.64	5.16	20.74
PQ3	LP1		1.64	5.16	20.74
PQ3	UP16	LP1	1.64	5.15	20.63
PQ3	UP8	LP1	1.64	5.07	20.57
PQ3	LP2		1.64	5.08	20.40
PQ3	UP32	LP2	1.64	5.08	20.40
PQ3	UP16	LP2	1.64	5.07	20.34
PQ3	LP3		1.64	5.01	20.16
PQ3	UP32	LP3	1.64	5.01	20.16
PQ3	UP16	LP3	1.64	5.01	20.16
PQ3	UP8	LP2	1.64	5	20.28
PQ3	UP8	LP3	1.64	4.95	20.10
PQ3	UP32	LP4	1.64	4.95	19.93
PQ3	LP4		1.64	4.95	19.93
PQ3	UP16	LP4	1.64	4.94	19.93
PQ3	UP8	LP4	1.64	4.89	19.87
PQ3	LP5		1.64	4.88	19.87
PQ3	UP32	LP5	1.64	4.88	19.87
PQ3	UP16	LP5	1.64	4.88	19.87
PQ3	UP8	LP5	1.64	4.83	19.81
PQ2	UP8		1.66	4.21	12.61
PQ2	UP16		1.66	4.33	12.61

binations are based on $PQ3$ which indicates that this is a solid single rule. The rest of the rules is dominated by the UP and LP families of rules which means that these are always good candidates for combination. The improvements can be found as well in the failure rate (under 22%), better accuracy (quality factors are almost at the level of LP) and better efficiency than comparably performing singles at the same time. This is not quite straightforward when compared with Fig. 5 because it suggests that worse rules would stop always before the better rules. This general view is still valid but at the level of individual runs there seems to be room for improvements and rule combinations can exploit them.

VI. CONCLUSIONS

We have performed an experimental survey of well established early stopping rules (ESR) proposed by the literature. Early stopping the training of a feedforward MLP is difficult and no optimal solution has been proposed until now. The aim is to provide a way of having a good classifier in a reasonable time. Since stopping at the right moment is still empirical and not well evaluated in practice, we have benchmarked 30 ESR and 2850 combinations of them on two standard public datasets: PROBEN1 and MNIST. We provided results in such a manner that they can be easily interpreted by the user according to his individual situation.

From our experimental results we propose to use PQ_2 or PQ_3 for most of the cases since they have a low fail rates and offers a good trade off between training duration and the quality of the achieved result.

In some specific cases LP and UP rules yield best network quality, but these rules should be only applied when this kind of behaviour has been repetitively observed already before a large scale MLP training.

If speed is critical then GSC or NR based rules should be used, where GSC delivers faster and where NR delivers better results. These methods stop early and have low fail rates meaning that they are robust.

We have shown that for better performance it is necessary to combine rules. Such combinations deliver faster with better accuracy and lower failure rates. However, hardly any rule can compete with cross-validation. Hence, if resources permit then cross-validation should be preferred.

Since ESR operate on plain curves and do not operate on underlying models or data it is not impossible that these results are transportable to other training algorithms or other neural models.

REFERENCES

[1] Rich Caruana and Ru Niculescu-Mizil, *An Empirical Comparison of Supervised Learning Algorithms*, In Proc. 23rd Intl. Conf. Machine learning (ICML'06), pp. 161-168, 2006

[2] D. Saad, *On-Line Learning in Neural Networks*, Cambridge University Press, 1998

[3] R. Ganguli and S. Bandopadhyay, *Neural network performance versus network architecture for a quick stop training application.* Application of Computers and Operations Research in the Mineral Industries, South African Institute of Mining and Metallurgy, 2003

[4] Rich Caruana and Alexandru Niculescu-Mizil and Geoff Crew and Alex Ksikes, *Ensemble selection from libraries of models*, in Proceedings of the twenty-first international conference on Machine learning, ICML'04, p. 18, ACM, New York, 2004

[5] Jurgen Branke, *Evolutionary algorithms for neural network design and training*, In Proceedings of the First Nordic Workshop on Genetic Algorithms and its Applications, pp. 145-163, 1995

[6] William Finno, Ferdinand Hergert and Hans Georg Zimmermann, *Improving model selection by nonconvergent methods.* Neural Networks Vol.6, pp 771-783, 1993.

[7] Lutz Prechelt, *Early stopping - but when*, Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2, pp. 55-69, Springer-Verlag, 1998

[8] Lutz Prechelt, *Automatic early stopping using cross validation: quantifying the criteria*, Neural Networks, pp. 761-767, 1998

[9] Mahesh S. Iyer and R. Russell Rhinehart, *A novel method to stop neural network training*, American Control Conference, 2000. Proceedings of the 2000, Vol. 2, pp. 929-933, Chicago, USA, 2000

[10] Thomas Breuel, *Pattern Recognition Engineering (PaREn)* http://sites.google.com/a/iupr.com/paren-project/Home

[11] Yann Le Cun, Leon Bottou and Genevieve Orr and Klauss-Robert Muller *Efficient BackProp*, http://www.research.att.com/ leonb/PS/lecun-98b.ps.gz, 1998

[12] Lutz Prechelt, *PROBEN1 - A Set of Neural Network Benchmarking Problems and Benchmarking Rules*, Technical Report 21/94, Faculty of Computer Science, University of Karlsruhe, 1994

[13] Yann LeCun, *The MNIST database*, http://yann.lecun.com/exdb/mnist/index.html

[14] S.E. Fahlman: *An empirical study of learning speed in back-propagation networks* CMU-CS-88-162, 1988

[15] Koch W., Schulz E.M., Wright R., Smith R.M., Lang S., et al. *What is a Ratio Scale?*, Rasch Measurement Transactions 9:4, p. 457, 1996

[16] Rajendra Bhatia, *The Logarithmic Mean*, Resonance, pp. 583-594, June, Bangalore, 2008

[17] http://archive.ics.uci.edu/ml/ *UCI Machine Learning Repository*